The Dissertation Committee for Ali Khodabakhsh
certifies that this is the approved version of the following dissertation:

Approximation Algorithms and Mechanism Design for Power Systems

Committee:

Evdokia Nikolova, Supervisor

Gustavo de Veciana

Sanjay Shakkottai

Hao Zhu

Swati Gupta

Approximation Algorithms and Mechanism Design for Power Systems

by

Ali Khodabakhsh

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2021

Dedicated to my family.

Acknowledgments

This thesis would not have been possible without the help and support of my family, friends, collaborators, and colleagues. I would like to express my deep and sincere gratitude to all those who made my journey at UT Austin memorable.

First, I would like to thank my advisor Prof. Evdokia Nikolova for her patience, motivation, enthusiasm, and continuous guidance throughout the past six years. It has been a great privilege and honor to work under her supervision, as I learned how to conduct research and to present my research as clearly as possible. I would like to thank my other committee members, Prof. Gustavo de Veciana, Prof. Sanjay Shakkottai, Prof. Hao Zhu, and Prof. Swati Gupta. I offer my sincere appreciation for their valuable and insightful feedback on my thesis. I would also like to thank Wireless Networking and Communications Group (WNCG) for providing a collaborative environment in which I could develop my research skills and broaden my perspective.

I would like to thank Simons Institute for the Theory of Computing at UC Berkeley, where I spent one semester in Real-Time Decision Making program. I would like to thank Evdokia Nikolova and Richard Karp for giving me the opportunity to meet such inspirational people there. I would especially like to thank Prof. Le Xie, Prof. Steven Low, Prof. Jannik Matuschke, Xinbo

Geng, Jimmy Horn, and Hassan Mortagy with whom I had the pleasure to brainstorm on exciting research problems.

During my PhD, I had the opportunity to collaborate with many brilliant and talented scholars. I am really grateful for their thought provoking discussions. I would especially like to thank Prof. Gonzalo Mateos, Prof. Samuel Taggart, Prof. Emmanouil Pountourakis, and Prof. Bo Li. I am also thankful for my lab mates and WNCG friends, Soumya, Ger, Thanasis, Orestis, Isidoros, Eftychia, Nithin, and Yutong. I would also like to take this opportunity to express my gratitude to ECE and WNCG administrative staff: Karen, Apipol, Jaymie, Melody, Barry, and Melanie.

My graduate studies would not have been as fun without the support and love of my friends. My special acknowledgment goes to Rasoul Shafipour who has been like a brother to me, and I have enjoyed his company in both intellectual collaboration and friendship. I would also like to thank Abolfazl, Mahsa, Kamyar, and Amir who made my life in Austin unforgettable. Finally, I would like to thank Mohammad for always making me laugh with his jokes, Ali for being the best roommate ever, and Parissa for proofreading this text.

Last but definitely not the least, I would like to thank my family for their support and their unconditional love. I am truly blessed to have wonderful parents who have always believed in me and helped me become the best person I can. I am also grateful to my sister for always being supportive, and for helping me achieve my goals. Thank you from the bottom of my heart.

Approximation Algorithms and Mechanism Design for Power Systems

Publication No. _____

Ali Khodabakhsh, Ph.D.
The University of Texas at Austin, 2021

Supervisor: Evdokia Nikolova

In real life, many combinatorial optimization problems are solved by simple algorithms (e.g., greedy, local search, thresholding, etc.) without any provable guarantees on the performance of the returned solution compared to the optimal one. Even though theoretical computer scientists have established strong results for the performance of such algorithms in diverse settings, it is not always easy to match the real life application to one of those well-studied problems. That said, we also have power systems which is a rich source of combinatorial optimization problems. The combinatorial nature arises whether it is turning switches on or off, assigning generation to different sources, scheduling charging or discharging of electric vehicles, splitting loads between different phases, or installing equipment at certain locations of the grid. On top of the combinatorial nature, these applications usually entail another level of complexity due to the nonlinearity of their objective and/or constraints.

In this Ph.D. dissertation, I will try to bridge theoretical computer science and power systems by utilizing tools from approximation algorithms and mechanism design to provide rigorous guarantees for three problems in power systems. Contributing to the intersection of these two fields is important to both push the state-of-the-art approximation algorithms by introducing new problems that cannot be solved with existing techniques, as well as provide methods for both scalable and rigorous solutions to problems arising in real-world power systems.

In the first problem, the goal is to minimize the power loss in a distribution network via changing the on/off status of switches in the network. Bringing to bear recent advances in submodular optimization, we derive performance bounds for a simple local-search algorithm that has been used in practice for a long time. We then focus on special properties of the grid that let us develop tailored algorithms with improved performance guarantees. Finally, we use our insights from our theoretical analysis to propose a general heuristic for the network reconfiguration problem that is orders of magnitude faster than existing methods in the literature, while obtaining comparable performance.

In the second problem, we propose a novel way to use electric vehicles as dynamic energy storage for supporting the power grid, by discharging them at designated times and locations of need. We show that the underlying scheduling problem is NP-hard and propose various approximation algorithms with different performance guarantees.

Finally in the last problem, we study a mechanism design problem for a system operator who wants to optimally assign the demand to different energy generators. We study this as a dynamic procurement auction, in which the generators may go out of business if they are not frequently picked. Since this will increase the electricity price in the long term, the system operator has to design a mechanism that minimizes the overall cost via selecting different sources frequent enough to maintain the competition. We show how to obtain optimal thresholding mechanisms via a greedy approach.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xiv**

List of Figures **xv**

Chapter 1. Introduction **1**

1.1 Motivation 1
1.2 Research Topics 3
1.3 Background 6
1.3.1 Approximation algorithms 6
1.3.2 Set functions and submodularity 7
1.3.3 Matroids 8
1.4 Thesis Outline 9

Chapter 2. Distribution Network Reconfiguration I: a Submodular Framework **11**

2.1 Introduction 12
2.2 Related Work 14
2.3 Problem Statement 17
2.4 Hardness Result 19
2.5 Submodular Framework 25
2.5.1 Proposed framework 25
2.5.2 Algorithm and analysis 29
2.6 Simulation Results 31
2.7 Conclusion 34

Chapter 3. Distribution Network Reconfiguration II: Algorithm Design **36**
3.1 Introduction . . . 37
3.2 Related Work . . . 41
3.3 Preliminaries . . . 43
3.4 A Randomized Iterative Edge-deletion Algorithm . . . 45
3.4.1 Iterative edge deletions and updates . . . 48
3.4.2 Performance of RIDE algorithm . . . 50
3.5 New Lower Bounds . . . 53
3.5.1 Shortest-path trees . . . 53
3.5.2 Cut-based lower bounds . . . 55
3.6 A 2-Approximation for Uniform Grid via MIN-MIN Algorithm 59
3.6.1 Lower bounds . . . 61
3.6.2 MIN-MIN algorithm . . . 63
3.6.3 Approximation factor for the MIN-MIN algorithm . . . 65
3.7 A Generalization of MIN-MIN: Layered-Matching . . . 70
3.8 Simulation Results . . . 71
3.9 Conclusion . . . 75

Chapter 4. Algorithms with Guarantees for Mobile Energy Storage Dispatch **76**
4.1 Introduction . . . 77
4.2 Related Work . . . 82
4.3 Mobile storage solution: Concept . . . 85
4.4 Problem Definition and Hardness . . . 89
4.4.1 Problem definition . . . 90
4.4.2 Hardness result . . . 92
4.5 Special Tractable Cases . . . 94
4.5.1 Zero charging time . . . 94
4.5.2 Single battery . . . 95
4.5.3 Constant number of batteries . . . 96
4.5.4 Homogeneous batteries . . . 97
4.6 Approximation Algorithms . . . 98

4.6.1 A $\frac{1}{3}$-approximation greedy algorithm 98
4.6.2 A $\frac{1}{2}$-approximation greedy algorithm 101
4.6.3 A $(1 - \frac{1}{e})$-approximation randomized algorithm 103
4.7 Simulation Results 105
4.7.1 Simulation setting 106
4.7.2 Presentation and analysis of the results 107
4.8 Conclusion 110

Chapter 5. Energy Procurement with Abandonment 111
5.1 Introduction 112
5.2 Related Work 126
5.3 Preliminaries 129
5.4 Mechanism 137
5.5 Optimality of Canonical Thresholds 143
5.6 Extensions 149
5.7 Conclusion 151

Appendices 153

Appendix A. Missing Proof from Chapter 2 154

Appendix B. Missing Proofs from Chapter 3 156
B.1 Proof of Laplacian Update (Lemma 3.4.3) 156
B.2 Proof of Energy Update (Lemma 3.4.4) 157
B.3 Stochastic Demands 158

Appendix C. Missing Material from Chapter 5 160
C.1 Bid-oblivious vs Bid-sensitive Mechanisms 160
C.2 Dealing with Ties 161
C.3 Distributional Assumptions 163
C.4 Non-monotone Example 167
C.5 Proof of Theorem 5.4.1 168

Bibliography 171

Vita **194**

List of Tables

2.1 Comparison of different DNR algorithms. 32

3.1 Comparing the optimality gap and running time of different algorithms on sparse 25×25 grids. 74

4.1 Empirical approximation ratios. 108

5.1 Different scenarios of the bids of two agents (Example 1) with respect to the threshold t. 138

List of Figures

2.1 Example of radial configuration in distribution grid. 15
2.2 Power flow variables. 17
2.3 Polynomial reduction. 21
2.4 Proof of Lemma 2.4.3. 24
2.5 Comparing losses from the simplified model with the exact values. 33
2.6 Rankings based on exact and approximate losses 34

3.1 The electrical flow on two graph instances, where all edges have unit resistance. 47
3.2 Lower-bound example on the performance of the shortest-path trees. 55
3.3 $n \times n$ grid with root at the top-left. 58
3.4 Snapshot of a real distribution network from a utility company in the US. 60
3.5 Example of the construction on the lower triangle of the grid. 64
3.6 Example of Min-Min algorithm. 65
3.7 An abstraction of a real-life distribution network through an instance of a sparsified 25×25 grid. 72
3.8 Performance of branch exchange and the mixed integer program initialized with the layered matching heuristic. 75

4.1 Thermal violations on a real distribution feeder. 78
4.2 Example of eliminating violations via mobile energy storage dispatch. 80
4.3 Example of thermal, voltage, phase imbalances, and power loss violations on a real distribution feeder. 86
4.4 Example of reducing 3D-matching problem to mobile energy storage dispatch. 93
4.5 Example of Randomized Rounding algorithm. 104
4.6 Performance ratio of the proposed approximation algorithms. . 109

5.1 Derivative of the cost with respect to the threshold for saving 3 agents. 122
5.2 Derivative of the cost with respect to a general threshold t_i . . 146

C.1 Regularity of a truncated normal distribution 164
C.2 Regularity of the polynomial distribution with exponent $p = 2$ 164
C.3 Regularity of the polynomial distribution with exponent $p = 1/2$ 165
C.4 Regularity of beta distribution 165

Chapter 1

Introduction

1.1 Motivation

Applications of combinatorial optimization are ubiquitous in real life, e.g., routing, matching, scheduling, etc. Unfortunately, many of these problems are NP-hard, meaning that there are no efficient algorithms to solve them exactly, unless P=NP. Therefore, we either have to resort to inefficient algorithms, or settle for approximate solutions. Inefficient algorithms can be useful when we deal with small instances of such problems, or when finding the optimal solution is much more important than the time spent to find it (for example, when we have to solve an instance just once, for a long-term decision). However, when we consider the large amounts of data and the importance of real-time decision making, for example in societal networks, there is no solution but to relax the optimality requirement. This is the topic of *approximation algorithms*, where we try to find "good enough" solutions in a time that is polynomial in the size of the input. In fact, the dramatic increase in the size of data has brought attention to linear and even sublinear algorithms nowadays. But in this dissertation, we follow the convention that an algorithm is efficient as long as its running time is polynomial in its input size.

Power systems is one of the areas where many combinatorial optimization problems appear. In the last decade, electricity networks have been facing a major transformation, from traditional operating methods to more automated, distributed, optimized algorithms. The combinatorial nature arises whether it is turning switches on or off, assigning generation to different sources, scheduling charging or discharging of electric vehicles, splitting loads between different phases, or installing equipment at certain locations of the grid. On top of the combinatorial nature, these applications usually entail another level of complexity due to the nonlinearity of their objective and/or constraints. In addition to the need for algorithm design to improve the efficiency of electricity networks, the operation of these networks has also changed from a unidirectional flow of energy (substations to users) to a bidirectional network where everyone can buy or sell electricity. Consequently, many *mechanism design* problems arise, as the users become decision-makers in this system.

In this Ph.D. dissertation, I will try to bridge theoretical computer science and power systems by utilizing tools from approximation algorithms and mechanism design to provide rigorous guarantees for three problems in power systems. Contributing to the intersection of these two fields is important to both push the state-of-the-art of approximation algorithms by introducing new problems that cannot be solved with existing techniques, as well as provide methods for both scalable and rigorous solutions to problems arising in real-world power systems.

1.2 Research Topics

The first problem we study in this thesis is the distribution network reconfiguration problem, in which the goal is to minimize the power loss in a distribution network via changing the on/off status of switches in the network. Since the distribution networks are acyclic and they should meet all demands, the search space becomes the set of all spanning trees of the underlying graph, which has exponential size in general. We show that this optimization problem is NP-hard. However, simple algorithms have been used to solve this problem for decades, without any performance guarantee. One of these simple algorithms is a local-search algorithm known as *branch exchange* [1, 2]. Bringing to bear recent advances in submodular optimization, we derive performance bounds for the branch exchange algorithm. We do this by showing that the distribution network reconfiguration problem is equivalent to minimizing a supermodular function subject to a matroid base constraint.

We then focus on special properties of the grid, that let us develop tailored algorithms with improved performance guarantees. For example, we show that if the underlying graph is a 2-dimensional grid and the edges have the same resistances, any shortest path tree would provide an $\mathcal{O}(\sqrt{n})$-approximation, where n is the number of nodes. If in addition to the edges, the loads are also uniform, we show how to achieve an $\mathcal{O}(1)$-approximation via an algorithm that we call Min-Min. Finally, we use our insights from our theoretical analysis to propose a general heuristic for the network reconfiguration problem that is orders of magnitude faster than existing methods in the

literature, while obtaining comparable performance.

In the next part of this dissertation, we propose a novel way to use electric vehicles (EVs) as dynamic energy storage for supporting the power grid, by discharging them at designated times and locations of need. As a benefit of this support, the utility companies can differ their expensive equipment upgrades, which could have been triggered by a few hours of violations in the absence of this concept. Hence in our storage dispatch concept, the utility sets up an agreement offering time and location varying rewards to a central planner for the dispatch of storage to utility designated locations of need. These locations have been identified by the utility company as stress points, via their periodic power flow analysis, and are equipped with discharging ports. The planner then computes an optimal dispatch of which batteries to utilize where and when to maximize its collected rewards, based on the varying battery availabilities. These batteries can be either from EV owners who provide their car in return of monetary compensation, or EV fleets owned/managed by cities or ride-sharing companies.

Towards the theoretical development of our state-of-the-art concept, we set up a mathematical model for the dispatch of a varying set of batteries at multiple time, location and price-varying distribution nodes. We study the computational complexity of this problem and show that this problem is strongly NP-hard. We then propose three approximation algorithms with worst-case performance guarantees of $1/3$, $1/2$ and $1-1/e \approx 0.63$. We further demonstrate the efficacy of our proposed algorithms via numerical simulations

and show that all three algorithms achieve an empirical performance between 82%-100%.

In the final part of this dissertation, we study a mechanism design problem for a system operator who wants to minimize the long-term cost of energy generation, by optimally allocating the demand to a set of energy generators. In a myopic approach, which is somehow similar to the common practice, the system operator picks the cheapest available sources everyday, until the demand is met. The problem with this approach is that in a lot of US markets, wind is typically the least expensive form of generation, thus it is favored by the current selection mechanism over conventional generation (nuclear, coal and gas). As a result, these conventional sources are gradually being driven out of business due to underuse, and the lack of competition can drive up the price in the future. As a result, this short-term cost-minimization approach yields a higher long-term cost.

In reality, a less competitive generator whose economic viability is threatened might be "saved" by the system operator, by entering a side contract that guarantees it sufficient allocation and payment to help it remain viable. Such contracts are currently done behind closed doors in an ad hoc way, including the system operator's decision which generators it considers critical. To improve system efficiency and transparency, we make a first step toward providing a framework for systematic allocation and payments that minimize cost over multiple periods.

We study a dynamic model of procurement auction in which the system

operator (buyer) repeatedly assigns the generation to a set of energy sources (sellers). We introduce the notion of "abandonment", meaning that a generator may go out of business and abandon the system if they do not receive enough payment. We show that under this model, the system operator may have to assign the generation to inefficient generators, in order to keep them alive, maintain the competition, and minimize the overall generation cost. We focus on threshold mechanisms for the system operator as a simple way to achieve incentive compatibility. We then consider the optimization problem of finding the optimal thresholds. We show that even though our objective function does not have the optimal substructure property in general, if the underlying distributions satisfy some regularity properties, the global optimal solution lies within a region where the optimal thresholds can be calculated with a simple greedy approach.

1.3 Background

In this section we present background material pertinent to the following technical chapters.

1.3.1 Approximation algorithms

Throughout this dissertation, we will work with approximation algorithms, defined as follows.

Definition 1.3.1 (**Approximation Algorithm** [3])**.** An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for

all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution. α is called the *performance guarantee* of the algorithm, also known as *approximation ratio* or *approximation factor*.

According to this definition, a $\frac{1}{2}$-approximation algorithm for a maximization problem is an algorithm which always achieves at least half of the optimal value. However, it is also common in the literature to measure the ratio of the optimal value by the approximate solution (i.e., the inverse of the above definition), which in that case the same algorithm would be called a 2-approximation. The notion used in each context should be clear by the type of problem (minimization or maximization) and approximation factor being greater/less than one.

1.3.2 Set functions and submodularity

Let V be a finite set, called the ground set. We use 2^V to denote the set of all subsets of V, called the power set. A set function $f : 2^V \mapsto \mathbb{R}$ is any function that maps the elements of the power set (i.e., the subsets of V) to real numbers. Function f is said to be submodular if it has the *diminishing returns* property, namely adding an element to a bigger set is less valuable than adding it to a smaller set.

Definition 1.3.2 (Submodularity). A set function $f : 2^V \mapsto \mathbb{R}$ with a ground set V is submodular if:

$$f(X \cup \{u\}) - f(X) \geq f(Y \cup \{u\}) - f(Y),$$

for every $X \subseteq Y \subseteq V$, $u \in V \backslash Y$.

Function f is said to be supermodular if $-f$ is submodular (or the above inequality holds in the other direction). Supermodularity captures an increasing returns property. A function is said to be modular if it is both submodular and supermodular.

Definition 1.3.3 (**Monotonicity).** A set function $f : 2^V \mapsto \mathbb{R}$ is said to be monotone increasing if $f(X) \leq f(Y)$ for any $X \subseteq Y \subseteq V$.

1.3.3 Matroids

Definition 1.3.4 (**Matroid** [4])**.** Let V be a finite set, and let $\mathcal{I}$ be a collection of subsets of V. The pair $\mathcal{M} = (V, \mathcal{I})$ is a matroid if the following conditions hold:

1. (Hereditary property) If $B \in \mathcal{I}$, then $A \in \mathcal{I}$ for all $A \subseteq B$.
2. (Augmentation property) If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists $v \in B \backslash A$ such that $A \cup \{v\} \in \mathcal{I}$.

A set $A \in \mathcal{I}$ is called an independent set. The collection $\mathcal{I}$ is called the set of independent sets of the matroid $\mathcal{M}$. A maximal independent set (an independent set that has maximum size) is a base of the matroid. It is easy to show that all the bases of a matroid have the same number of elements. The following theorem introduces an example of a matroid over the edges of an undirected graph.

Theorem 1.3.1 (**Cycle Matroid** [4]). *Let $G(V, E)$ be an undirected graph. Define the set T to be the collection of all subsets of E that form a forest (i.e., the subset is acyclic). In other words, $A \in T$ iff $A \subseteq E$ and edges in A do not form a cycle. Then $\mathcal{M} = (E, T)$ is a matroid called the cycle matroid of graph G (also known as graphic matroid).*

1.4 Thesis Outline

The remainder of this dissertation is organized as follows.

Chapter 2 presents the distribution network reconfiguration problem through the lens of submodular optimization. In this chapter, we show that this problem is equivalent to minimizing a supermodular set function subject to a graphic matroid base constraint. Using recent developments in submodular optimization, we pioneer to provide the first performance bound for the well-known branch exchange algorithm that has been used in practice for a long time. Missing proofs of this chapter are provided in Appendix A. This chapter concludes with numerical experiments that compare various algorithms proposed for this problem in the literature.

Chapter 3 revisits the distribution network reconfiguration problem with an algorithm design approach. The main contribution of this chapter is to provide novel lower bounds and corresponding approximation factors for various settings ranging from $\min\{\mathcal{O}(m-n), \mathcal{O}(n)\}$[1] for general graphs, to

[1] m, n are the number of edges and vertices in the graph, respectively.

$\mathcal{O}(\sqrt{n})$ over grids with uniform edge resistances, and $\mathcal{O}(1)$ for grids with uniform edge resistances and demands. The chapter concludes with extensive numerical simulations that demonstrate the efficacy of our proposed algorithms. Missing proofs of this chapter are provided in Appendix B.

Chapter 4 presents our new concept of using electric vehicles as mobile energy storage systems. The chapter starts with the description of our concept with its potential real-world impact, theory-to-concept and other implementation considerations. We also propose in this chapter various approximation algorithms to bring this concept to fruition. The chapter concludes with numerical experiments that compare the performance of our proposed algorithms.

Finally, Chapter 5 studies the dynamic procurement auction with abandonment, motivated by the problem of minimizing the cost of energy generation via a set of energy sources that have private viability constraints. We prove in this chapter how the optimal mechanism can be achieved by a greedy algorithm. Missing proofs of this chapter are provided in Appendix C.

Chapter 2

Distribution Network Reconfiguration I: a Submodular Framework

Distribution network reconfiguration (DNR) is a tool used by operators to balance line load flows and mitigate losses. As distributed generation and flexible load adoption increases, the impact of DNR on the security, efficiency, and reliability of the grid will increase as well. Today, heuristic-based actions like branch exchange are routinely taken, with no theoretical guarantee of their optimality. In this chapter, we consider loss minimization via DNR, which changes the on/off status of switches in the network. The goal is to ensure a radial final configuration (called a spanning tree in the algorithms literature) that spans all network buses and connects them to the substation (called the root of the tree) through a single path. We prove that the associated combinatorial optimization problem is strongly NP-hard and thus likely cannot be solved efficiently. We formulate the loss minimization problem as a supermodular function minimization under a single matroid basis constraint, and use existing algorithms to propose a polynomial time local search algorithm for the DNR problem at hand and derive performance bounds. We show that our algorithm is equivalent to the extensively used branch exchange algorithm, for which, to the best of our knowledge, we pioneer in proposing a

theoretical performance bound. Finally, we use a 33-bus network to compare our algorithm's performance to several algorithms published in the literature.[1]

2.1 Introduction

Distribution networks are usually built as interconnected mesh networks, but are normally configured (via switches) and operated as radial networks (i.e. trees, in graph theoretic terms), to simplify overload protection [6]. The entire network can be thought of as a forest consisting of rooted trees. Each tree consists of a substation (root) and a number of customers (users) that are serviced via so-called distribution feeders (distribution lines starting at the substation). Switches located throughout the network allow dynamic reconfiguration of the distribution network through switching operations; the opening or closing of a switch corresponds to the removal or addition of an edge, respectively.

The goal of distribution networks is to deliver the power from substations to users, but notably, substantial losses of up to 13% occur as electric power flows over distribution lines [7]. As a result, *Distribution Network Reconfiguration* is a major tool focusing on the dynamic identification of a spanning tree that optimizes a performance measure such as load flow balancing or total line loss minimization. We select the latter, namely the minimization of losses

[1]The content of this chapter has appeared in the proceedings of the 51st Hawaii International Conference on System Sciences (HICSS) [5]. The author of this dissertation made the primary technical contribution to this work.

for a given hourly load flow, as the objective of the reconfiguration problem. Similar issues in meshed transmission networks have been addressed in the literature recently (see [8] and references therein).

The main contributions of this chapter are the following:

1. We prove that the DNR problem is strongly NP-hard. We do this through a polynomial reduction from 3-PARTITION problem, which is defined in Section 2.4 (see [9] for more details). To the best of our knowledge, the computational hardness of this problem has not been studied so far.

2. We formulate the DNR problem as a supermodular minimization problem subject to a single matroid basis constraint (see Section 1.3 for the definitions of supermodularity and matroid). Supermodularity is motivated by the fact that losses are quadratic in the current flowing over each branch of the distribution network. Furthermore, the matroid basis constraint ensures the radial structure and guarantees that all the buses are connected to the substation.

3. We observe that the local search algorithm for solving the supermodular minimization problem is equivalent to the well-known *branch exchange* algorithm. Hence, we obtain the first theoretical result on why the branch exchange algorithm performs well in practice.

The proposed submodular framework sheds some light on the algorithmic structure of the optimization problems in distribution networks. Although

in this chapter we are mostly providing a theoretical justification for an existing heuristic, this helps us to provide new algorithms in the next chapter. It is also evident in other lines of work in energy systems (see [10–12] for example) that the theoretical study of such problems can help to either find new algorithms or improve the efficiency of existing ones.

2.2 Related Work

DNR has been studied extensively in the literature. One of the most common heuristic algorithms is the *branch exchange* suggested by Civanlar *et al.* [1] and implemented by Baran and Wu [2], who considered loss minimization and load balancing objectives. Starting from a feasible tree configuration, the branch exchange algorithm transfers some loads in each iteration by (i) closing an open switch to create a loop in the network, followed by (ii) opening one of the closed switches in that loop to arrive at another feasible solution with a lower cost. The algorithm terminates when no further improvements are possible. This algorithm has been used as a benchmark against different DNR algorithms with the 12.6kV network of Fig. 2.1 employed for numerical comparisons.

An improved branch exchange algorithm was proposed by Miguez *et al.* [14] who tried to expand the space of available changes in the local search, hence eliminating some local minima of the standard algorithm. The idea of improved branch exchange is to investigate improvement from a pair of exchanges, once there is no improvement by a single branch exchange. Peng

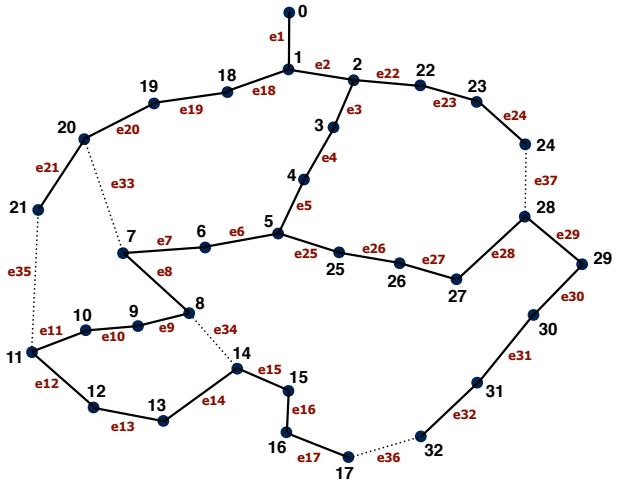

Figure 2.1: 33-bus network [13].

and Low [15] proposed an algorithm to do each step of branch exchange efficiently by solving only 3 optimal power flow equations (OPF), regardless of the size of the network. Their algorithm helps to find the best switch to open in order to minimize any convex increasing cost function, assuming that an open switch has already been closed. These improvements still provide no theoretical guarantee on the output of the branch exchange algorithm.

Unlike the branch exchange algorithm that maintains a tree structure during its execution, there are other heuristic algorithms that start with the meshed network (obtained by closing all the tie switches) or the disconnected network (obtained by opening all the switches) and proceed to open/close switches one by one until a radial configuration is achieved [16–18]. Shirmohammadi and Hong [17] proposed one such algorithm that starts with the meshed network and proceeds with iterations that open the switch with the smallest current. No theoretical performance guarantees have been obtained

for this algorithm.

For small networks such as the 33-bus example of Fig. 2.1, the global optimal configuration can be discovered by brute-force enumeration. An efficient enumeration approach proposed in [19], lists all the spanning trees in a clever way that generates each tree exactly once, and calculates losses by adjusting the losses of the previous spanning tree. The drawback of this method is that it is not practical for larger networks, since a network has exponentially many spanning trees [20].

The joint DNR and OPF problem was considered in [13] using Benders decomposition to decompose the global problem to master and slave subproblems. The master level determines the binary variables by solving a mixed-integer non-linear program using CPLEX. The slave level solves the OPF non-linear program using the CONOPT solver. Again, solving integer programs is computationally intractable for large networks.

Many other approaches like genetic algorithms [21, 22], particle swarm optimization [6], ant colony algorithms [23], artificial neural networks [24, 25], etc. have been utilized to solve this problem. A survey of different algorithms for the DNR problem can be found in [7]. What is conspicuously missing in all these previous works is a rigorous theoretical performance guarantee.

To close this gap, we consider a submodular approach to the DNR problem. Since switching binary decisions render DNR a non-linear combinatorial optimization problem, additional structure like submodularity or supermodu-

larity enables finding an approximate solution efficiently.

2.3 Problem Statement

In this section we present the power flow equations and employ some simplifying assumptions to model the problem in graph theoretic terms. We model the distribution network as a graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of buses (nodes) and $\mathcal{E}$ is the set of lines (undirected edges). We assume that a single substation is located at node 0, and the other nodes are load buses with given active and reactive power demands (p_i, q_i), for all $i \in \mathcal{N}\backslash\{0\}$. We are looking for a spanning tree rooted at bus 0 (i.e., a tree that connects all the loads to the root through a single path) which minimizes the total resistive loss.

Letting $V_i = |V_i|e^{\mathbf{i}\theta_i}$ represent the complex voltage at bus i, we adopt the relaxed branch model of [15, 26] that allows us to ignore the phase angles of voltages and currents in radial networks. Let $Z_e = R_e + \mathbf{i}X_e$ be the impedance of line $e \in \mathcal{E}$. We also use $S_{ij} = P_{ij} + \mathbf{i}Q_{ij}$ to express the branch power flow from bus i to bus j, and I_{ij} to express the current from bus i to j. A summary of our notation is depicted in Fig. 2.2.

Assumption 2.3.1 ([15, A2])**.** Voltage variation across the distribution network can be neglected. Using per unit (p.u.) representation, we assume that

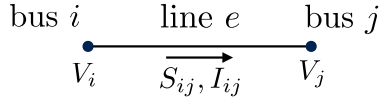

Figure 2.2: Power flow variables.

$|V_i| = 1$ p.u. for all nodes $i \in \mathcal{N}$.

This assumption is realistic since in practice voltage at every bus is kept within an allowable range such as $(0.95, 1.05)$ p.u., and impacts losses (the objective function of DNR) at a smaller order of magnitude than different spanning trees. Moreover, this assumption does not change significantly the ordering of spanning trees based on the associated line losses.

Assumption 2.3.2. The impact of line losses on line flows is negligible relative to the power demands at the buses of the network.

This assumption implies that the power flow on each line $e \in \mathcal{E}$ is almost equal to the total demand of the buses that are receiving power through that line. Specifically, for a given spanning tree, if we denote the set of successors of an edge $e \in \mathcal{E}$ by $\text{succ}(e)$, then we have:

$$P_e = \sum_{i \in \text{succ}(e)} p_i \quad \text{and} \quad Q_e = \sum_{i \in \text{succ}(e)} q_i, \tag{2.1}$$

where p_i and q_i are the active and reactive power demands at bus i. Note that by P_e we mean the power flowing on line e in the direction from the root of the tree to the leaves (parent to child). In Section 2.6, we verify the validity of these assumptions in detail.

If we denote the loss of line $e = \{i, j\} \in \mathcal{E}$ by L_e, then we have:

$$L_e = R_e \times |I_{ij}|^2. \tag{2.2}$$

In addition, in the relaxed model we have:

$$|V_i|^2|I_{ij}|^2 = P_{ij}^2 + Q_{ij}^2. \tag{2.3}$$

Combining (2.1), (2.2), and (2.3) with Assumption 2.3.1 implies that:

$$L_e = R_e \left[\left(\sum_{i \in \text{succ}(e)} p_i \right)^2 + \left(\sum_{i \in \text{succ}(e)} q_i \right)^2 \right].$$

Given a spanning tree (ST) we can sum up the line losses L_e over all the edges of the tree to find the total loss. Thus, the optimal reconfiguration problem with the goal of loss minimization can be written as the following optimization problem:

$$\min_{ST} \quad \sum_{e \in ST} R_e \left[\left(\sum_{i \in \text{succ}(e)} p_i \right)^2 + \left(\sum_{i \in \text{succ}(e)} q_i \right)^2 \right], \tag{P1}$$

where the minimization is over all the spanning trees of $\mathcal{G}(\mathcal{N}, \mathcal{E})$.

2.4 Hardness Result

In this section we prove that the DNR problem is strongly NP-hard in general by a reduction from the 3-Partition problem [9]. A computational hardness result is more powerful when it is derived for a more restricted setting—since the hardness implication then holds for any generalization of the setting. Here we derive a hardness result for the special case of unit demands, where the objective function of the optimization problem (P1) reduces to a simpler function that is just counting the number of successors. In particular

we make the following assumptions:

$$\begin{aligned} R_e &= 1 \qquad \forall e \in \mathcal{E}, \\ p_i &= 1 \qquad \forall i \in \mathcal{N} \backslash \{0\}, \\ q_i &= 0 \qquad \forall i \in \mathcal{N} \backslash \{0\}. \end{aligned}$$

Under these assumptions, the optimization problem (P1) reduces to the following problem:

$$\min_{ST} \quad \sum_{e \in ST} (\text{number of successors of } e \text{ in } ST)^2. \tag{2.4}$$

Although these assumptions may not be realistic, they transform the problem into an explicit combinatorial problem (without any power flow variable or parameter) and help us to analyze the computational complexity of the reconfiguration problem. The resulting complexity applies then to more general and realistic settings, as mentioned above.

We show that even the unit-demand case is strongly NP-hard. We prove this by a reduction from the 3-Partition problem defined as follows.

Definition 2.4.1. (3-Partition) In the 3-Partition problem we have a multiset of $k = 3m$ integers summing to mB with each integer strictly between $B/4$ and $B/2$. The task is to partition these numbers into m triplets each with a sum of B.

It is well known that in the 3-Partition problem, deciding whether a given multiset can be partitioned into balanced triplets or not, is strongly

NP-complete, i.e., it is NP-complete even if the numbers are bounded by a polynomial in the length of the input [9].

Theorem 2.4.1 (**Hardness result**). *Distribution network reconfiguration problem (P1) is strongly NP-hard.*

Proof. We propose a polynomial reduction from the 3-PARTITION problem to the unit-demand case of reconfiguration problem (P2). Given an instance of the 3-PARTITION problem we build an instance of the reconfiguration problem such that the optimal spanning tree reveals the answer to the 3-PARTITION problem (if it exists). Given $k = 3m$ integers $\{a_1, a_2, ..., a_k\}$, we construct a network as shown in Fig. 2.3. There is a root r, m nodes $u_1, ..., u_m$ connected to the root, $k = 3m$ nodes $v_1, ..., v_k$ each connected to all of u_i's (thus v_i's and u_j's form a complete bipartite graph) and for each v_i we have $a_i - 1$ nodes connected to it.

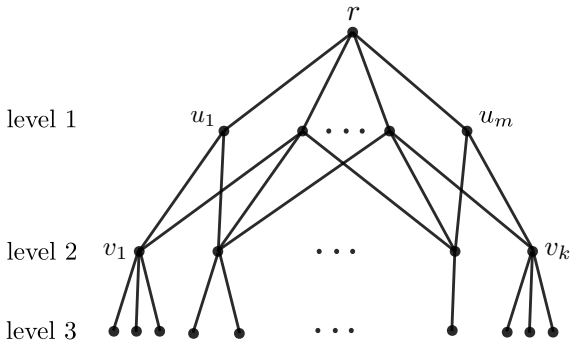

Figure 2.3: Polynomial reduction.

Lemma 2.4.3 below proves that all the lines between the root r and the u_j's are part of the optimal tree. Moreover, all the lines between levels

two and three appear in every spanning tree, so the only choices are on the lines between levels one and two. In particular, we have to connect each v_i to exactly one u_j, i.e., make one of m choices.

When we connect node v_i to node u_j, the corresponding edge gets a cost of a_i^2, since there are $a_i - 1$ nodes in level three and the edge has a_i successors including v_i. This cost is independent of the choice of u_j, so the total cost for the edges between levels one and two is the same for all the spanning trees. The cost to be minimized is thus the total cost of the edges between root r and the u_j's. Let S_j be the set of indices of the children of u_j, i.e.,

$$S_j = \{i \mid (u_j, v_i) \in \text{Tree}\},$$

then the cost related to edge (r, u_j) is:

$$C_{(r,u_j)} = \left(1 + \sum_{i \in S_j} a_i\right)^2,$$

where 1 counts for the node u_j itself. Now the total cost of the spanning tree is:

$$\begin{aligned} C &= \sum_{j=1}^{m} C_{(r,u_j)} + \sum_{i=1}^{3m} a_i^2 + \sum_{i=1}^{3m} (a_i - 1) \\ &= \sum_{j=1}^{m} \left(1 + \sum_{i \in S_j} a_i\right)^2 + \sum_{i=1}^{3m} a_i^2 + (mB - 3m). \end{aligned} \tag{2.5}$$

As mentioned earlier, the second and third terms are constants since they are independent of the choice of the spanning tree. Using the fact that the S_j's

are disjoint and $\cup_{j=1}^{m} S_j = \{1, ..., k\}$, we also have:

$$\sum_{j=1}^{m} \left(1 + \sum_{i \in S_j} a_i\right) = m + \sum_{j=1}^{m} \sum_{i \in S_j} a_i = m + \sum_{i=1}^{3m} a_i = m + mB = m(B+1). \tag{2.6}$$

Lemma 2.4.2. *The minimum of $\sum_{i=1}^{n} x_i^2$ given that $\sum_{i=1}^{n} x_i = C$ for a constant $C \in \mathbb{R}$ is achieved when $x_i = C/n$ for all $1 \leq i \leq n$.*

By Lemma 2.4.2 and (2.6), the minimum possible cost of (2.5) is obtained when:

$$1 + \sum_{i \in S_j} a_i = B + 1 \qquad \forall j \in \{1, ..., m\},$$

and the optimal value is:

$$C_{min} = m(B+1)^2 + \sum_{i=1}^{3m} a_i^2 + (mB - 3m).$$

Note that this optimal cost is achieved when a_i's are partitioned into m subsets with sum B, but there is no restriction on the size of S_j's. This means that node u_j can have any number of v_i's connected to it, while in the 3-Partition problem we want to partition the a_i's into m triplets. The property $B/4 < a_i < B/2$ ensures that this minimum can only be achieved when $|S_j| = 3$ for all j. If for any j' we have $|S_{j'}| > 3$, then we get:

$$\sum_{i \in S_{j'}} a_i > 4 \times \frac{B}{4} = B,$$

and the partition cannot be balanced. Similarly if $|S_{j'}| < 3$, then we get:

$$\sum_{i \in S_{j'}} a_i < 2 \times \frac{B}{2} = B.$$

In conclusion, the algorithm for the unit-demand case finds the tree corresponding to the 3-PARTITION answer (if it exists), and if it outputs some unbalanced tree, this means that the 3-PARTITION does not exist. If each a_i is bounded by a polynomial in k, the constructed network has polynomial number of nodes, hence any polynomial time algorithm for the unit-demand case provides a pseudo-polynomial time algorithm for the 3-PARTITION problem which is not possible unless $P = NP$. □

Lemma 2.4.3 proves the only remaining part of the hardness proof.

Lemma 2.4.3. *With uniform line resistances ($R_e = R, \forall e \in \mathcal{E}$), the optimal tree includes all the edges adjacent to the root.*

Proof. We prove this by contradiction. Assume that we have an optimal tree which does not choose edge (r, u) as shown in Fig. 2.4 on the left. Let W be the total load weight of subtree connected to u (including u). Since we have a tree, this subtree is connected to the root through another node v. Node v may have other children and also may be connected to the root via one or more

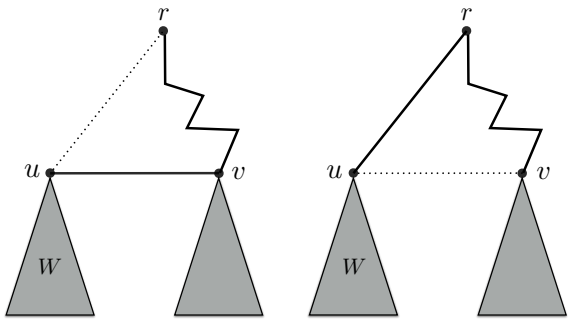

Figure 2.4: Proof of Lemma 2.4.3.

edges. Now we claim that this tree cannot be optimal since we can exchange edge (u, v) with edge (r, u) and improve the objective value as shown in the right tree. To see this, note that both edges (u, v) in the left tree and (r, u) in the right tree have costs RW^2, but the exchange of (u, v) with (r, u) decreases the load on all the edges of the path from r to v by W, hence decreasing the total cost. This contradicts the optimality of the first tree. □

Note that the unit-demand case is a special case of Lemma 2.4.3.

2.5 Submodular Framework

In the previous section we showed that DNR is strongly NP-hard, but if we find some additional structure such as submodularity or supermodularity in the problem, we may be able to provide approximation algorithms, which provides a rigorous worst-case performance guarantee. Here we show that the DNR problem has this structure.

2.5.1 Proposed framework

Considering the formulation of DNR (P1), the optimization problem is over all the spanning trees of the original graph. We would like to encode the two properties of "being a tree" and "touching all the vertices of the graph" into a set of constraints. In order to do this, we need to define a set of variables as follows. These variables also help to determine the successors of an edge in any arbitrary tree.

- For any edge $e \in \mathcal{E}$ we define a variable x_e that indicates if that edge is included in the tree or not (the number of variables is equal to the number of lines in the distribution network).

- Corresponding to any variable x_e, where $e = \{i, j\}$, we also define y_{ij}^k and y_{ji}^k for all $k \in \mathcal{N}$, which indicate the position of node k compared to edge $e = \{i, j\}$. If there is a simple path from i to k including $\{i, j\}$, then $y_{ij}^k = 1$ and if there is a simple path from j to k including $\{i, j\}$, then $y_{ji}^k = 1$. In other words, $y_{ij}^k = 1$ means that edge $\{i, j\}$ is chosen and j is on the path from i to k. If $x_e = 0$, then both y_{ij}^k and y_{ji}^k are zero.

The following theorem, inspired by the integer programming formulation for the minimum spanning tree problem [27, 28], explains how we use these variables to characterize the spanning trees explicitly.

Theorem 2.5.1 (**Feasible set characterization).** *There is a one-to-one correspondence between the spanning trees of* $\mathcal{G}(\mathcal{N}, \mathcal{E})$ *and the feasible set specified by the following set of constraints:*

$$\sum_{e \in \mathcal{E}} x_e = n - 1 \tag{2.7}$$

$$y_{ij}^k + y_{ji}^k = x_e \qquad \forall e = \{i, j\} \in \mathcal{E}, \forall k \in \mathcal{N} \tag{2.8}$$

$$x_e + \sum_{k \neq i,j} y_{ik}^j = 1 \qquad \forall i, j \in \mathcal{N} : e = \{i, j\} \in \mathcal{E} \tag{2.9}$$

$$x_e, y_{ij}^k, y_{ji}^k \in \{0, 1\} \qquad \forall e = \{i, j\} \in \mathcal{E}, \forall k \in \mathcal{N} \tag{2.10}$$

If we write the total loss as a function of the binary variables above, we end up with an integer program formulation of (P1). For a given spanning tree T (equivalently, a feasible set of values for the binary variables), and an edge $e = \{i, j\} \in T$, the variables y_{ij}^k and y_{ji}^k induce a partition of the vertices $\mathcal{N}$ into two sets which are exactly the two connected components of the tree obtained by removing $\{i, j\}$. The set that does not include the root (assuming vertex 0 is the root), is the set of successors of e in T. In other words, if $y_{ji}^0 = 1$, and succ(e) is the set of its successors, then we have:

$$\text{succ}(e) = \{k \in \mathcal{N} : y_{ij}^k = 1\}.$$

Note that $y_{ji}^0 = 1$ is not an additional assumption, since the edges are not directed, and hence for the edges in the tree, one of the pairs (i, j) or (j, i) satisfies this condition.

Using this new description of successors, we can rewrite the objective function as:

$$\sum_{e \in ST} R_e \left[\left(\sum_{i \in \text{succ}(e)} p_i \right)^2 + \left(\sum_{i \in \text{succ}(e)} q_i \right)^2 \right] = \sum_{i,j:\{i,j\} \in \mathcal{E}} R_{ij} y_{ji}^0 \left[\left(\sum_{k \in \mathcal{N}} y_{ij}^k p_k \right)^2 + \left(\sum_{k \in \mathcal{N}} y_{ij}^k q_k \right)^2 \right], \quad (2.11)$$

where the inner summations are over all nodes, but the y_{ij}^k's guarantee that we only count the successors, and the term y_{ji}^0 outside guarantees that we calculate each edge of the tree exactly once and in the correct direction with respect to the root.

So, the following optimization problem is equivalent to (P1):[2]

$$\begin{aligned} \min \quad & \sum_{\{i,j\}\in\mathcal{E}} R_{ij} y^0_{ji} \left[\left(\sum_{k\in\mathcal{N}} y^k_{ij} p_k \right)^2 + \left(\sum_{k\in\mathcal{N}} y^k_{ij} q_k \right)^2 \right] \\ s.t. \quad & (2.7), (2.8), (2.9), (2.10). \end{aligned} \tag{2.12}$$

Now we show that (2.12) is equivalent to a supermodular minimization problem with a single matroid basis constraint. The objective function (2.11) is not supermodular over $\mathcal{E}$, but we create a similar set function that is supermodular and is equal to (2.11) when constraints (2.7–2.10) hold (i.e., for spanning trees). The following is a corollary of Theorem 1.3.1 which shows that the feasible set in (2.12) is indeed a matroid basis constraint.

Corollary 2.5.2. *Assuming that graph $\mathcal{G}$ is connected, the bases of the cycle matroid $\mathcal{M}$ are the spanning trees of $\mathcal{G}$, which all have cardinality $|\mathcal{N}| - 1$. Therefore, constraints (2.7–2.10) are equivalent to a single matroid basis constraint on $\mathcal{E}$.*

Now we introduce the supermodular set function over $\mathcal{E}$. For any $A \subseteq \mathcal{E}$, we define:

$$f(A) = \sum_{\{i,j\}\in\mathcal{E}} R_{ij} z^0_{ji} \left[\left(\sum_{k\in\mathcal{N}} z^k_{ij} p_k \right)^2 + \left(\sum_{k\in\mathcal{N}} z^k_{ij} q_k \right)^2 \right] \tag{2.13}$$

The only difference between (2.11) and (2.13) is that we replaced the y^k_{ij}'s with z^k_{ij}'s, and z^k_{ij} is defined similar to y^k_{ij} except that it can be any non-negative integer (compared to $0, 1$) and it counts the number of paths in A starting

[2] we use $\sum_{\{i,j\}\in\mathcal{E}}$ instead of $\sum_{i,j\in\mathcal{N}:\{i,j\}\in\mathcal{E}}$ for simplicity.

with $\{i, j\}$ and going to k. Clearly, for spanning trees there cannot be more than one path between any arbitrary pair of vertices, therefore $z_{ij}^k = y_{ij}^k$ and this implies the equality of (2.11) and (2.13) when constraints (2.7–2.10) hold.

Theorem 2.5.3 (**Supermodularity**). *Objective function (2.13) is a supermodular set function over $\mathcal{E}$, provided that the p_i's and q_i's are non-negative.*

Proof. See Appendix A. □

2.5.2 Algorithm and analysis

In the previous section we showed that the DNR problem (2.12) is equivalent to a supermodular minimization problem subject to a single matroid basis constraint. Unless $P = NP$, it is not possible to approximate the minimum of a supermodular function within any factor [29], in contrast with the related problem of maximizing a submodular function which admits a constant factor approximation algorithm [30]. We adapt the approximation algorithm for the submodular maximization problem under matroid constraints, proposed by Lee *et al.* [30], to solve the DNR problem, but we have to convert the supermodular function to a non-negative submodular function (by negating and shifting). This conversion affects the multiplicative approximation guarantee, as shown in Theorem 2.5.4. The algorithm, which is based on local search, is described in Algorithm 1.

The algorithm starts with an arbitrary spanning tree T. Then at each

Algorithm 1 Distribution Network Reconfiguration for Loss Minimization

Input: Configuration $\mathcal{G}(\mathcal{N}, \mathcal{E})$, bus demands (p_k, q_k), line resistances (R_{ij}), ϵ.
Output: Spanning tree for minimizing the total loss.
Initialize T with an arbitrary spanning tree.
while 1 **do**
    if there exist $e \in \mathcal{E}\backslash T$ and $e' \in T$ such that $(T\backslash\{e'\})\cup\{e\}$ is a spanning tree and $f((T\backslash\{e'\}) \cup \{e\}) < (1-\epsilon)f(T)$ **then**
        $T \leftarrow (T\backslash\{e'\}) \cup \{e\}$
    else
        break
    end if
end while

iteration, it looks for two edges $e \in \mathcal{E}\backslash T$ and $e' \in T$ such that swapping those two edges makes another spanning tree with loss at most $(1-\epsilon)f(T)$. If such a pair exists, it updates T and repeats the exchange process, otherwise the algorithm terminates and outputs the locally optimal spanning tree.

Theorem 2.5.4 (**Performance guarantee).** *Let T_{alg} be the output of Algorithm 1, and T^* be the optimal spanning tree, i.e., $T^* = argmin\{f(T) : T \subseteq \mathcal{E}, T$ is a spanning tree$\}$. Let $M = f(\mathcal{E})$, which is an upper bound on $f(A)$ for all $A \subseteq \mathcal{E}$, then:*

$$M - f(T_{alg}) \geq \left(\frac{1}{6} - \epsilon\right)\big(M - f(T^*)\big). \tag{2.14}$$

Proof. This is a corollary of [30, Theorem 22], which provides a $(\frac{1}{6} - \epsilon)$-approximation algorithm for maximizing any non-negative submodular function over bases of a matroid $\mathcal{M}$.[3] Here we use $M - f(A)$ as the non-negative

[3]That theorem requires $\mathcal{M}$ to have at least two disjoint bases. We can solve this (if nec-

submodular function, and the spanning trees are the bases of the cycle matroid discussed in Theorem 1.3.1. □

Even though Algorithm 1 is based on the local search approximation algorithm for maximizing non-monotone submodular functions [30], it is equivalent to the branch exchange heuristic algorithm which has been used since the late 1980s [2]. This establishes that Theorem 2.5.4 provides the first proof of a performance bound, and hence a performance guarantee for the branch exchange algorithm.

2.6 Simulation Results

Table 2.1 shows the results of our experiments on the 33-bus network of Fig. 2.1. The parameters of the network can be found in [6]. All the active and reactive power demands are positive for this network as assumed in Theorem 2.5.3. The simulations have been done by using the MATPOWER package in MATLAB [31]. The results show that in this case, our submodular approach finds the globally optimal configuration, which was found in [19] (by enumerating all 50751 spanning trees). In [2], 2 different approximate power flow methods with different accuracies have been used and we also believe that there are inconsistencies regarding the parameters of the network in the

essary) by adding dummy edges with very high resistances (to make sure that the algorithm never selects them). Moreover, their algorithm performs another local search which allows deletion of elements, but that run yields the empty set in our case (due to the monotonicity), hence does not apply to the DNR problem.

Table 2.1: Comparison of different DNR algorithms on the 33-bus network of Fig. 2.1

Algorithm	Method	Open Lines	Loss (kW)
Proposed	Submodular Local Search	$7, 9, 14, 32, 37$	139.552
Morton, Mareels [19]	Brute-Force	$7, 9, 14, 32, 37$	139.552
Gomes *et al.* [16]	Greedy on Mesh Network	$7, 9, 14, 32, 37$	139.552
Khodr, Martinez [13]	Benders Decomposition	$7, 9, 14, 32, 37$	139.552
Wu *et al.* [23]	Ant Colony	$7, 9, 14, 28, 32$	139.976
Shirmohammadi [17]	Optimal Current Pattern	$7, 10, 14, 32, 37$	140.279
Baran, Wu [2][4]	Branch Exchange	$11, 28, 31, 33, 34$	146.832
Initial Configuration		$33, 34, 35, 36, 37$	202.670

literature[4]; that is why results reported in [2] differ from what we obtained by Algorithm 1. Clearly, the output of the local search algorithms depends on the initialization. We used the initial configuration (Fig. 2.1) as the initial spanning tree in our simulation. Further, to check the robustness with respect to the initial tree, we repeated the simulations with 1000 random initial trees, all of which ended with the same optimal solution. In order to check the validity of our assumptions (see the problem formulation in Section 2.3), we compare the losses of spanning trees as measured in (P1) with the exact losses obtained from MATPOWER. The result is shown in Fig. 2.5. The blue line is the exact loss curve where the spanning trees are sorted in the order of increasing total loss. The red dots also show the loss for each tree obtained from the simplified model. We observe that the approximate loss is generally increasing, which means that it can be used in the local search algorithm. In fact performing the local search with either exact loss or approximate loss

[4]The resistance of the branch between bus 6 and bus 7 is 0.7114Ω in [2], but 1.7114Ω in [6]. We used the latter value in all our simulations.

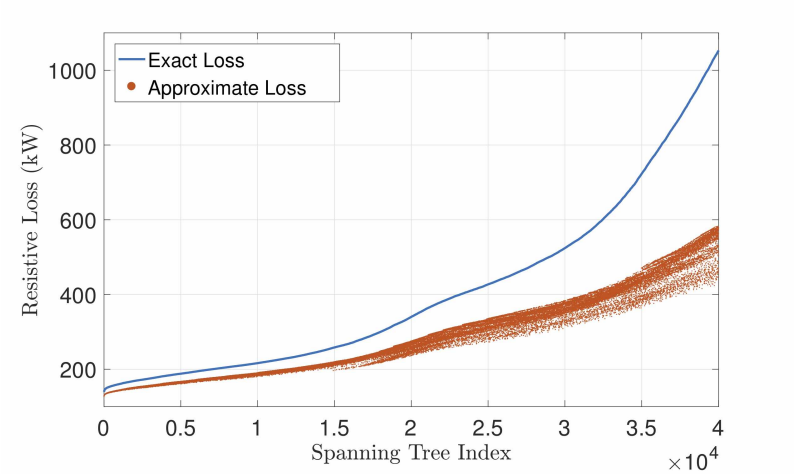

Figure 2.5: Comparing losses from the simplified model with the exact values.

results in the same globally optimal tree, reported in Table 2.1. As expected, approximate loss estimates are less accurate for trees with higher losses, since the resistive losses approach the order of magnitude of load demands in such networks (hence contradicting Assumption 2). On the other hand, for trees with smaller losses (which are indeed the target of our optimization problem) the simplified loss approximates the exact loss very well.

Fig. 2.6 also compares the rank of the top 5000 spanning trees based on the exact and approximate losses. Ideally, we would like the simplified losses to preserve the rankings (which would result in a $y = x$ line in this plot). We observe that no single spanning tree faces a significant change in its ranking.

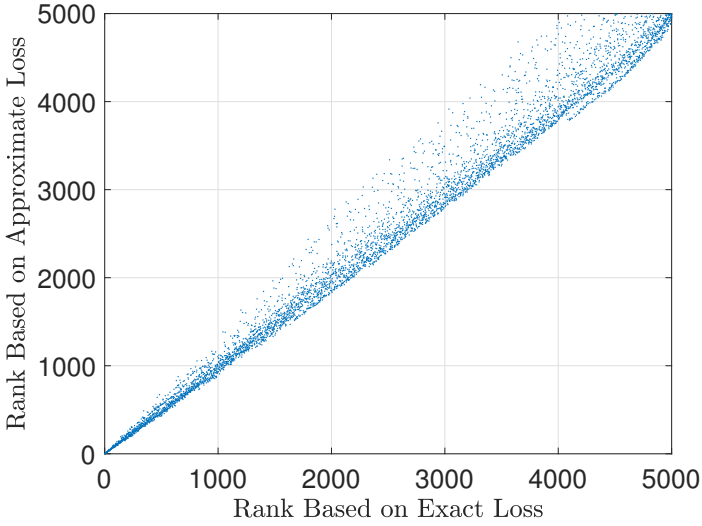

Figure 2.6: Rankings based on exact and approximate losses for the best 5000 spanning trees.

2.7 Conclusion

In this chapter, we studied the distribution network reconfiguration problem for loss minimization through a submodular optimization approach. We proved that this problem is NP-hard even if the demands and the line resistances are all equal to one. We formulated this problem as a supermodular minimization problem subject to a matroid basis constraint. We then used the algorithm for maximizing non-monotone submodular functions under matroid constraints, to give a polynomial time algorithm for the DNR problem with a performance guarantee. The algorithm was equivalent to the branch exchange algorithm that was known previously, but for which no theoretical guarantees were available. By discovering a submodular structure in the problem, we pioneered the derivation of a performance bound on the branch exchange

algorithm.

In the next chapter, we try to propose direct combinatorial algorithms for the distribution network reconfiguration problem, compared to using the existing results from submodular optimization. This allows us to prove multiplicative constant factor approximation results, compared to the performance bound of Theorem 2.5.4 which includes the upper bound M.

Chapter 3

Distribution Network Reconfiguration II: Algorithm Design

In this chapter, we revisit the *distribution network reconfiguration* problem from Chapter 2 which seeks to find a spanning tree T such that the loss of the electrical flow over T is minimized. The tree requirement on the support of the flow is motivated by operational constraints in electricity distribution networks. The bulk of existing results on convex optimization over vertices of polytopes and on the structure of electrical flows do not easily give guarantees for this problem, while many heuristic methods have been developed in the power systems community as early as 1989 [32]. The main contribution of this chapter is to give the first provable approximation guarantees for the network reconfiguration problem. We provide novel lower bounds and corresponding approximation factors for various settings ranging from $\min\{\mathcal{O}(m-n), \mathcal{O}(n)\}$ for general graphs, to $\mathcal{O}(\sqrt{n})$ over grids with uniform resistances on edges, and $\mathcal{O}(1)$ for grids with uniform edge resistances and demands. To obtain the result for general graphs, we propose a new method for (approximate) spectral graph sparsification, which may be of independent interest. Using insights from our theoretical results, we propose a general heuristic for the network reconfiguration problem that is orders of magnitude faster than existing methods

in the literature, while obtaining comparable performance.[1]

3.1 Introduction

Electricity distribution and transmission networks have been a rich source of non-convex problems with combinatorial structure that have helped discover limitations of, as well as advance, the theory of discrete and continuous optimization [34–36]. In this chapter, we revisit the *distribution network reconfiguration* problem which seeks to find a rooted tree such that the energy of the electrical flow over the tree is minimized.[2]

For simplicity of presentation, we assume in this chapter that all the demands are real numbers, denoted by d_i. However, this is without loss of generality and all our approximation guarantees hold for the more general case of $d = p + \mathbf{i}q$. In the general case, the objective function can be decomposed into two additive parts, in which one is only a function of real demands (p), and the other is only a function of the reactive part (q). The reason why our approximation guarantees still hold is that our proposed solutions would guarantee the same approximation factor for both parts of the objective. Moreover, here we assume that the demands are known parameters, while we discuss the extensions of our results to the stochastic demand case in Appendix B.3.

Given an undirected graph $G = (V, E)$ ($|V| = n$, $|E| = m$), root $r \in V$,

[1]The content of this chapter has been published in the Mathematical Programming journal (MAPR) [33]. The author of this dissertation made major technical contributions to this work.

[2]We use energy and loss interchangeably.

resistances $r_e > 0$ for each edge $e \in E$, real demands $d_i \geq 0$ for each node $i \in V \backslash \{r\}$ supplied by the root node, thus $d_r = -\sum_{i \in V \backslash \{r\}} d_i$, the *distribution network reconfiguration* problem can be formulated as the following flow problem:

$$\min \quad \mathcal{E}(f) := \sum_{e \in E} r_e f_e^2 \tag{3.1}$$

$$\text{subject to} \quad \sum_{e \in \delta^+(i)} f_e - \sum_{e \in \delta^-(i)} f_e = d_i, \ \forall i \in V, \tag{3.2}$$

$$\text{support}(f) \text{ is acyclic}, \tag{3.3}$$

where $\delta^+(v)$ and $\delta^-(v)$ denote the sets of incoming and outgoing edges of v (after fixing an arbitrary orientation on the edges), f is any feasible flow satisfying the demands (3.2)[3], the support of f is constrained to be acyclic (and therefore, a tree rooted at r) (3.3), and the objective (3.1) is to minimize the energy of the flow. While there may be more than one feasible flow satisfying the demands in general, an electrical flow minimizes the energy subject to meeting the demands. Moreover, given an r-rooted tree, there is a unique flow f on the tree that satisfies the demands: $f_e = \sum_{i \in \text{succ}(e)} d_i$, where $\text{succ}(e)$ is the set of nodes that connect to the root through e. Also, note that any r-rooted tree can be augmented to be a spanning tree in the graph at no additional cost.[4] Therefore, the network reconfiguration problem (3.1) can

[3]Here we use a simplified linear flow model, similar to [37]. In reality, power flow equations are nonlinear and result in non-convex optimization problems [38–41]. We refer the reader to a recent survey on relaxations and approximations of power flow equations [36]. In contrast, here we aim to relax the non-linearity of the power flow model, and instead focus on the combinatorial aspect of the optimization problem in (3.1).

[4]By contracting the support of the existing flow and adding edges with 0 flow to construct

then be equivalently written as:

$$\min_{T \in \mathcal{T}} \quad \sum_{e \in T} r_e \left(\sum_{i \in \mathrm{succ}(e)} d_i \right)^2 , \tag{P2}$$

where $\mathcal{T}$ is the set of all spanning trees of G.

In this chapter, we provide a provable approximation guarantee for the network reconfiguration problem (P2). We delve deeper into the problem structure to construct novel lower bounds and give new ways of graph sparsification while maintaining original edge-resistances. Our theoretical insights also lead to a significant improvement in computation. The main contributions of this chapter are the following:

(a) *Flow Relaxation and new* RIDE *algorithm:* We first relax the spanning tree constraint, reducing the problem to finding a minimum energy electrical flow; we call this the *flow relaxation*. In Section 3.4, we construct instances that have a gap between the energy of the optimal tree and the flow relaxation of the order $\Theta(\sqrt{n}/\log n)$ over grid graphs and $\Theta(\Delta)$ in general graphs, where n and Δ are the number of nodes and maximum degree in the graph (which can be linear in n) respectively. Further, we propose a randomized iterative edge-deletion algorithm, RIDE, that sparsifies the graph by deleting edges sampled according to a specific probability distribution dependent on the *effective* resistances of remaining edges. We show that this method can guarantee $\mathcal{O}(m-n)$ approxi-

a spanning tree.

mation with respect to the flow relaxation. This technique of sparsifying graphs may be of independent interest.

(b) *New Lower Bounds:* We next exploit the combinatorial structure of the graph to obtain novel lower bounds in Section 3.5. We first show that any shortest-path tree, with respect to resistances, gives an $\mathcal{O}(n)$ approximation. Though a simple argument, this is already better than the $O(m^2 - m)$ bound using [42] and the approximation using RIDE for dense graphs. Second, we show that we can improve this approximation factor by finding a laminar family of cuts. In particular, for certain grid instances that have uniform edge resistances, a selection of laminar cuts gives an $\mathcal{O}(\sqrt{n})$-approximation. These results serve as preliminaries for our second combinatorial algorithm MIN-MIN explained next.

(c) *Constant-factor approximation using new* MIN-MIN *algorithm:* Real-life distribution networks often resemble subgraphs of mesh-like networks. For such networks, like grid graphs with n nodes and uniform demands, the above mentioned techniques, are able to provide only an $\Omega(\sqrt{n}/\log n)$ approximation. Motivated by this, we construct a purely combinatorial algorithm MIN-MIN in Section 3.6, that finds a specific shortest path tree over an $n \times n$ grid with uniform resistances and a root at one of the corners of the grid. We show that MIN-MIN gives a $(2+\mathcal{O}(\frac{1}{\ln(n)}))(\frac{d_{\max}}{d_{\min}})^2$ approximation when the demands are in $[d_{\min}, d_{\max}]$. In particular, for uniform demands MIN-MIN gives an asymptotic 2-approximation.

(d) *Layered Matching Heuristic and Computational Results:* Inspired by the algorithmic ideas in MIN-MIN, we next propose a layered matching heuristic, LM-Heuristic. This heuristic can be used to find approximate solutions in the most general setting, without assumptions on the structure of network, resistances or demands. Using computational experiments over randomly sparsified grid networks, we find that LM performs very well in practice. In addition, the algorithms proposed in this chapter take orders of magnitude less time than the best known heuristic for this problem, the branch exchange heuristic, while obtaining comparable performance. For example, on 25×25 grid instances with sparsification probability $p = 0.2$, the mean time taken by LM is 1.35 seconds, whereas the mean time it takes the Branch Exchange heuristic to attain the same cost as LM is around 10 hours. We believe this improvement will be crucial in enabling system operators to reconfigure distribution networks more frequently in practice.

3.2 Related Work

Network reconfiguration problem requires minimizing a quadratic function over the vertices of the general flow polytope (Theorem 7.4 in [43]). In particular, if all the resistances are uniform, then the problem is equivalent to finding a vertex with the smallest Euclidean norm, which is known to be NP-hard (see for example Lemma 4.1.4 in [44]). For general polytopes, the best approximation one can hope to get in polynomial time, for minimizing a

general strongly convex function over integral points in a polytope is $O(n^2 - n)$ where n is the dimension of the polytope [42].

Besides distribution networks, switching problems have also been studied in electricity transmission networks, such as optimal transmission switching [45–47] and maximum transmission switching [48]. Despite the hardness of these problems [49, 50], Grastien *et al.* [48] show how to achieve a 2-approximation for maximum transmission switching on cacti graphs. However, for transmission networks there is no requirement on the support to be acyclic, thus making our problem structurally very different from those above.

In the network reconfiguration problem, relaxing the tree constraint (3.3) results in the well-studied problem of computing electrical flows as they uniquely minimize energy [51–53]. However, existing results in spectral sparsification [54, 55], that sparsify a graph without changing the energy much, do not extend to our setting since they change the resistances on the remaining edges to compensate for edge deletions. Many existing heuristics involve iterative edge-deletion (e.g., [32]) using the electrical flow values in the resultant graph, but offer no provable guarantees.

Another relevant approach in the electric flows literature, entails *low stretch trees*. Given a weighted graph G, a low-stretch spanning tree T is a spanning tree with the additional property that it approximates distances between the endpoints of any edge in G. In particular, the *stretch* of an edge $e = (u, v)$ is the ratio of the (unique) shortest path distance between u and

v in T to r_e (the weight of edge e in G).[5] Furthermore, the *total stretch* of T is defined as the sum of the stretch of all edges in G. Kelner *et al.* [52] show that for any tree, the gap between the energy of the flow in that tree and the flow in the original graph, is at most the total stretch of that tree. Naturally, one may wonder if there exists a low value for the (total) stretch such that all graphs have a spanning tree with that stretch. The answer to that question is unfortunately no. Abraham and Neiman [57] show that one can construct a spanning tree T for any connected graph with total stretch at most $\mathcal{O}(m \log n \log \log n)$ in near-linear time (Theorem 2.11 in [52]); this bound is tight up to an $\mathcal{O}(\log \log n)$ factor because Alon et. al [59] show that the total stretch is $\Omega(m \log n)$ for certain graph instances. Thus, this implies that the energy cost of T is at most $\tilde{\mathcal{O}}(m)$ times that of the original graph. We improve upon this approximation result using our RIDE algorithm.

3.3 Preliminaries

Relaxing the support constraint in the network reconfiguration problem reduces the problem to computing an electrical flow, that we refer to as the *flow relaxation.* Electrical flows have been shown to be efficiently computable in near-linear time [37, 51, 53, 60], used to speed up the computation of maximum

[5] We follow the definition of Elkin *et al.* [56], but this definition slightly differs in the denominator from others given in the literature. Abraham and Neiman [57] and Abraham *et al.* [58] define the stretch as $\frac{d_T(u,v)}{d_G(u,v)}$, where d_G is the shortest-path metric on G with respect to the edge weights. Note that these definitions are equivalent if the edge weights are uniform.

flow [60], and even bound the integrality gap of the asymmetric traveling salesman problem (ATSP) [61]. We give a brief review of preliminaries on electrical flows, that will be useful in understanding the results in this paper.

Let $G = (V, E)$ be a connected and undirected graph with $|V| = n$, $|E| = m$. Let $B \in \mathbb{R}^{n \times m}$ be the vertex-edge incidence matrix upon orienting each edge in E arbitrarily. Let R be an $m \times m$ diagonal resistance matrix where $R_{e,e} = r_e$. A key matrix that will play a fundamental role in the analysis of our algorithms is the weighted Laplacian $L := BCB^T$, where $C = R^{-1}$. It is well known that if G is connected, the only vector in the nullspace of the Laplacian L is the all-ones vector $\mathbf{1}$.

In what follows, we will invert the Laplacian matrix using the Moore-Penrose pseudoinverse denoted by $L^\dagger$; in which case $LL^\dagger$ is a projection matrix that projects onto the span of the columns of L, which we denote by $\mathrm{im}(L)$. Let $b \in \mathbb{R}^n$ be the feasible node-demand vector. The optimality conditions of the flow relaxation problem imply the existence of a vector of potentials on the nodes (dual variables) $\phi \in \mathbb{R}^n$ such that $\phi = L^\dagger b$ (this is well-defined since $\mathbf{1}^T b = 0$) and the optimal electrical flow $f = CB^T\phi$. Using these facts, one can show that the optimal energy $\mathcal{E}(f) = R^T f R = \phi^T L \phi = b^T \phi = b^T L^\dagger b$. For any pair of vertices u, v the effective resistance $R_{\text{eff}}(u, v)$ is the energy of sending *one* unit of electrical flow from u to v. In particular, for any vertex $u \in V$, if we let $\mathbb{1}_u \in \mathbb{R}^n$ be the characteristic vector of u, then $R_{\text{eff}}(u, v) = \chi_{uv}^T L^\dagger \chi_{uv}$, where $\chi_{uv} = \mathbb{1}_v - \mathbb{1}_u$ and can be thought of as the demand vector in this case. While the above notation suffices for our purposes, for more background on

electrical flows, we refer the reader to comprehensive book chapters in [62] and [63]. We next discuss our novel randomized iterative edge-deletion algorithm, RIDE, that rounds a fractional point (i.e., minimum energy flow, relaxing the support constraint in (3.1)) in the flow polytope, while maintaining a provable increase in the energy.

3.4 A Randomized Iterative Edge-deletion Algorithm

The key idea of our RIDE algorithm is to delete edges iteratively following a specific probability distribution, while maintaining the graph connectivity, until the resultant graph is a spanning tree. This is done as follows: sample an edge e at random to delete from the graph with probability p_e proportional to $1 - c_e R_{\text{eff}}(e)$, where $c_e = 1/r_e$ is the conductance of an edge. The normalization constant of this probability distribution is known to be $\sum_{e \in E}(1 - c_e R_{\text{eff}}(e)) = m - (n-1)$ (see for instance [64]). Intuitively, the quantity $c_e R_{\text{eff}}(e)$ fully characterizes the graph's ability to *reconfigure* the flow upon deleting edge e (as we show later in Lemma 3.4.4). In particular, the smaller the $c_e R_{\text{eff}}(e)$ (thus, the larger probability of deleting the edge), the better the graph's ability to re-route the flow of edge e upon deleting the edge, without significantly increasing the energy. Moreover, this sampling procedure ensures that connectivity is maintained, since $p_e = 0$ for any bridge edge e, as in such a case we have $R_{\text{eff}}(e) = r_e$. To implement this algorithm efficiently, we will show that the resultant graph Laplacian and effective resistances can be efficiently updated after every deletion. We give the complete description

Algorithm 2 Randomized iterative deletion (RIDE) algorithm

Input: A graph $G_0 = (V, E)$ and resistances $r : E \to \mathbb{R}_{++}$.

1: **for** $k = 0, \ldots, m - n$ **do**
2: Compute $R_{\text{eff}}^{(k)}(e)$ for all $e \in E$, i.e., the effective resistance for each e in G_k
3: Sample edge $e \in E$ according to probabilities $p_e^{(k)} = (1 - c_e R_{\text{eff}}^{(k)}(e))/(m - k - (n-1))$
4: $G_{k+1} \leftarrow G_k \setminus \{e\}$
5: **end for**

Return: Spanning tree $T = G_{m-n+1}$

of the algorithm in Algorithm 2.

We will also show that RIDE gives an $\mathcal{O}(m - n)$ approximation factor in expectation with respect to the cost of the flow relaxation. This is the first approximation guarantee for randomized edge deletion heuristics (such as [32]), to the best of our knowledge.

Theorem 3.4.1. *The randomized iterative edge-deletion algorithm,* RIDE*, gives an $\mathcal{O}(m - n)$ approximation in expectation with respect to the cost of the flow relaxation $\mathcal{E}(f_G)$ on the given graph G. In particular, $\mathbb{E}[\mathcal{E}(f_T)] \leq \mathcal{E}(f_G)(m - n + 2)$.*

Note that the above theorem implies that for planar graphs, RIDE gives an $O(n)$ approximation (since number of edges is linear in the number of nodes). In general, it seems one cannot obtain better than $\Omega(n)$ performance using rounding of electrical flows, unless the flow relaxation is strengthened using new inequalities. For instances with maximum degree Δ, the gap from the flow relaxation can be $\Omega(\Delta)$. Consider a graph with two nodes r, t (r is the root, and t is the only node with positive demand, say 1 unit) with

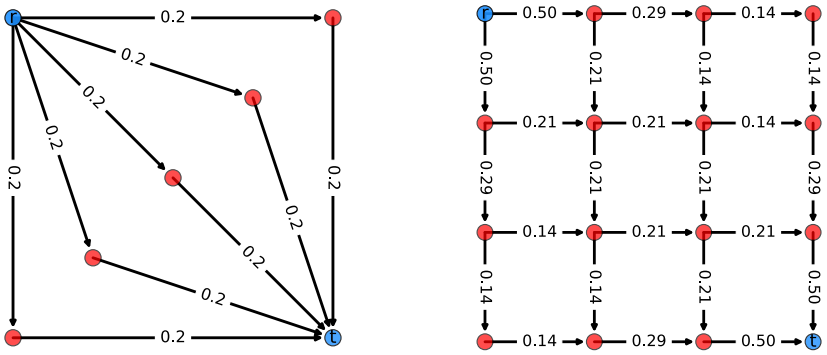

Figure 3.1: The electrical flow on two graph instances, where all edges have unit resistance, the root is node r, node t has unit demand and all other nodes have zero demand.

$n-2$ edge-disjoint paths between them; see Figure 3.1 (left). In this case, the electrical flow sends $1/\Delta$ units of flow along each of the Δ disjoint r-t paths. The energy of the electric flow is then $\Theta(1/\Delta)$, however, the energy of the optimal spanning tree is 2. Therefore, the gap from flow relaxation can be as large as $\Omega(n)$, since Δ can be as large as $\Theta(n)$.

Moreover, the gap of the optimal tree compared to the flow relaxation can be large even for graphs with small Δ. Consider a $\sqrt{n} \times \sqrt{n}$ grid (with n nodes) where the root r is in the top left corner, all edges have unit resistances, the node in the bottom right corner, call it t, has a demand of one, and all other nodes (excluding the root) have zero demand; see Figure 3.1 (right) for an example (here $\Delta = 4$). Recall that the potential drop between r and t on sending *one* unit of current from r to t is equal to the effective resistance $R_{\text{eff}}(r,t)$ between r and t. Hence, using Ohm's Law this implies that the energy of the electrical flow is equal to $R_{\text{eff}}(r,t)$. In this instance, it is known that

(see Proposition 10.11 in [65])

$$\frac{1}{2}\log\sqrt{n} \leq R_{\text{eff}}(r,t) \leq 2\log\sqrt{n}.$$

Furthermore, the cost of any optimal tree is $2(\sqrt{n}-1)$, since any r-t path has hop-length $2(\sqrt{n}-1)$ and the path from r-t can be grown into a spanning tree without incurring any additional cost. Combining these two facts, we get the gap for grid instances is $\Theta(\sqrt{n}/\log\sqrt{n}) = \Theta(\sqrt{n}/\log n)$.[6]

Even existing work on analyzing the stretch of trees that has been used to bound the energy of a tree with respect to the flow relaxation [52], does not give compelling approximation bounds, since there exist instances with stretch at least $\Omega(m\log n)$ [59]. We next present details of our sparsification approach followed by the analysis of RIDE.

3.4.1 Iterative edge deletions and updates

Before we delve into the proof for the performance of RIDE, we discuss one of the main components in designing and analyzing the algorithm: determination of how the energy of the electrical flow changes after deleting an edge from the graph. As mentioned in preliminaries, electrical flows are fully determined by the Laplacian and its pseudoinverse. Thus, to determine how electrical flows change upon edge deletions, we first obtain a closed form expression for how the Laplacian pseudoinverse changes from one iteration to

[6] We will improve upon this factor in Section 3.6.2.

the next. The following lemma provides a formula for updating pseudoinverses following rank-one updates.

Lemma 3.4.2 (Theorem A.70 in [66]). *Suppose that $A \in \mathbb{R}^{n\times n}$ is symmetric matrix, $u, v \in \mathbb{R}^n$ are vectors in $\mathrm{im}(A)$, and $1 - v^T A^\dagger u \neq 0$. Then*

$$(A - uv^T)^\dagger = A^\dagger + \frac{A^\dagger uv^T A^\dagger}{1 - v^T A^\dagger u}. \tag{3.4}$$

Now if the rank-one update corresponds to deleting an edge, we can show the following update rule.

Lemma 3.4.3. *Let L be the weighted Laplacian (weighted by conductances) of a connected graph $G = (V, E)$ and $L^\dagger$ be the Moore-Penrose pseudoinverse of L. Let $G' = G \setminus \{e\}$ be the graph obtained by deleting an edge $e \in E$ that does not disconnect G. Then the Moore-Penrose pseudoinverse of the weighted Laplacian of G' is:*

$$(L')^\dagger = L^\dagger + \frac{L^\dagger \chi_e c_e \chi_e^T L^\dagger}{1 - c_e \chi_e^T L^\dagger \chi_e}. \tag{3.5}$$

Proof. See Appendix B.1. □

We can now use Lemma 3.4.3 to obtain a closed form expression for how the energy increases from one iteration to the next. Suppose that we have performed k iterations and deleted k edges from the original graph G to get a modified connected graph $G_k = (V, E_k)$ with $m - k$ edges. Let B_k, and L_k respectively be the incidence and Laplacian matrices of G_k. Also, Let $f_k(e)$ and $R_{\mathrm{eff}}^{(k)}(e)$ respectively be the electrical flow and effective resistance for edge

e in iteration G_k. Now, in the $(k+1)^{\text{th}}$ iteration we wish to delete another edge e from G_k to obtain a connected graph G_{k+1}. We want to determine how much $\mathcal{E}(f_{k+1})$ changes compared to $\mathcal{E}(f_k)$.

Lemma 3.4.4. *Assume G_k is connected and $G_{k+1} = G_k \setminus \{e\}$ is obtained by deleting an edge e from G_k such that G_{k+1} is also connected. Then $\mathcal{E}(f_{k+1})$ can be recursively obtained from $\mathcal{E}(f_k)$ using*

$$\mathcal{E}(f_{k+1}) = \mathcal{E}(f_k) + \frac{r_e f_k(e)^2}{1 - c_e R_{\mathit{eff}}^{(k)}(e)}. \tag{3.6}$$

Proof. See Appendix B.2. □

3.4.2 Performance of RIDE algorithm

We are now ready to present the proof for Theorem 3.4.1.

Proof. Let $G_k = (V, E_k)$ be the resultant graph in iteration k after k edges have been deleted, and let $\mathcal{E}(f_k)$ be the energy of the electrical flow in G_k. In particular, $\mathcal{E}(f_0)$ denotes the cost of the flow relaxation. We will first show that

$$\mathbb{E}[\mathcal{E}(f_k)] \leq \mathcal{E}(f_0) \left(\frac{m - n + 2}{m - k - n + 2} \right) \tag{3.7}$$

and that G_k remains connected throughout the iterations of the algorithm by induction on k ($0 \leq k \leq m - n + 1$). The base case when $k = 0$ holds trivially. For the inductive step assume the result holds true for all iterations $k < m - n + 1$. Now consider iteration $k + 1$ of the RIDE algorithm applied to the graph G_k. Recall that we sample an edge $e \in E_k$ at random to delete

from the graph with probability $p_e^{(k)} = \frac{1-c_e R_{\text{eff}}^{(k)}(e)}{(m-k)-(n-1)}$. Since by the induction hypothesis G_k is connected, and this sampling procedure does not disconnect the graph, G_{k+1} is also connected. Moreover, we could use Lemma 3.4.4 to compute the expected increase in energy upon deleting edge e (which does not disconnect the graph), where

$$\mathcal{E}(f_{k+1}) = \mathcal{E}(f_k) + \frac{r_e f_k(e)^2}{1 - c_e R_{\text{eff}}^{(k)}(e)}.$$

To that end, let $Z^{(k)}$ be a random variable denoting the increase in the energy from iteration k to $k+1$. In particular, upon deleting an edge $e \in E_k$ that does not disconnect G_k, we have $Z_e^{(k)} = r_e f_k(e)^2/(1 - c_e R_{\text{eff}}^{(k)}(e))$. Also, let $E'_k = \{e \in E_k \mid p_e^{(k)} > 0\}$ be the set of edges that are not bridges. Then, since $E'_k \subseteq E_k$ we have

$$\begin{aligned}\mathbb{E}[Z^{(k)} \mid E_k] = \sum_{e \in E'_k} Z_e^{(k)} p_e^{(k)} &= \sum_{e \in E'_k} \frac{r_e f_k(e)^2}{1 - c_e R_{\text{eff}}^{(k)}(e)} \frac{1 - c_e R_{\text{eff}}^{(k)}(e)}{(m-k)-(n-1)} \\ &\leq \frac{\mathcal{E}(f_k)}{m-k-n+1},\end{aligned}$$

which in turn gives, using iterated expectations:

$$\mathbb{E}[Z^{(k)}] = \mathbb{E}[\mathbb{E}[Z^{(k)} \mid E_k]] \leq \frac{\mathbb{E}[\mathcal{E}(f_k)]}{m-k-n+1}.$$

Therefore,

$$\begin{aligned}\mathbb{E}[\mathcal{E}(f_{k+1})] = \mathbb{E}[\mathcal{E}(f_k) + Z^{(k)}] &\leq \mathbb{E}[\mathcal{E}(f_k)] \left(1 + \frac{1}{m-k-n+1}\right) \\ &\leq \mathcal{E}(f_0) \left(\frac{m-n+2}{m-k-n+2}\right) \left(\frac{m-k-n+2}{m-k-n+1}\right) \\ &= \mathcal{E}(f_0) \left(\frac{m-n+2}{m-(k+1)-n+2}\right)\end{aligned}$$

where we used the induction hypothesis in the second inequality. This concludes the induction and proves the correctness of the algorithm. Lastly, to obtain the final expected cost of the algorithm, we use (3.7) with $k = m-n+1$ to obtain $\mathbb{E}[\mathcal{E}(f_{m-n+1})] \leq \mathcal{E}(f_0)(m-n+2)$, as claimed. Since the cost of the optimal spanning tree is lower bounded by the cost of the flow relaxation, $\mathcal{E}(f_0)$, the result follows:

$$\mathbb{E}[\mathcal{E}(f_T)] \leq \mathcal{E}(f_G)(m-n+2),$$

where $\mathcal{E}(f_T)$ is the energy of the tree T returned by RIDE and $\mathcal{E}(f_G)$ is the energy of the flow relaxation. □

Note that the above approximation factor is with respect to the *flow relaxation*, which is a loose lower bound as discussed previously. For planar graphs in particular, the gap between the optimal solution and the flow relaxation can be $\Omega(n)$ (e.g., when the maximum degree is linear), thus in some sense RIDE is optimal up to a constant factor for the planar case. However, this does not preclude the possibility of obtaining a better lower bound and approximation using electrical flows and this remains an open question. To strengthen the lower bound and consequently obtain better approximation factors, we proceed by exploiting the combinatorial structure in certain graphs as well as demand scenarios.

3.5 New Lower Bounds

As discussed in Section 3.4, relaxing the spanning tree constraint can lead to a weak lower bound, mainly because we allow the demand of a single node v to be delivered via multiple paths from root r to node v. A natural question at this point is if we can strengthen the flow relaxation by exploiting the tree constraints. We first answer this question by accounting for the minimum loss each node v creates to get connected to the root, in the absence of all other nodes, and show how to use this lower bound to achieve an n-approximation algorithm. Next, we consider cuts in a graph to lower bound the energy of an optimal tree by considering the demand it separates. We show that by constructing a laminar family of cuts, one can derive a new lower bound by accounting for the loss of a spanning tree which is balanced over all these cuts. We then use this lower bound to get a $\sqrt{n}$-approximation for grid graphs.

3.5.1 Shortest-path trees

The major source of hardness in (P2) is the quadratic loss function, which introduces a cross-term for any two nodes that share an edge on their path to the root. If the loss was a linear function of the flow, and in the absence of cross-terms, the problem would decompose into n disjoint problems, which could be solved via shortest path trees. Note that, however, we can relate the quadratic loss to the linear case as follows:

$$\sum_{i \in \text{succ}(e)} d_i^2 \leq \left(\sum_{i \in \text{succ}(e)} d_i \right)^2 \leq n \times \sum_{i \in \text{succ}(e)} d_i^2, \tag{3.8}$$

where the first inequality is by the non-negativity assumption, and the second one is due to the Cauchy-Schwarz inequality. Now looking at d_i^2 as the new demand of each node i, our quadratic loss lies between the two linear objectives, for which we have to solve the following optimization problem: $\min_{T \in \mathcal{T}} \sum_{e \in T} \left[r_e \times \sum_{i \in \text{succ}(e)} d_i^2 \right]$. It is easy to show that the shortest-path tree rooted at node r (with respect to edge resistances, r_e) solves this optimization problem[7], which immediately implies the following result.

Theorem 3.5.1. *The shortest-path tree (with respect to resistances) rooted at r is an n-approximation solution for problem* (P2)*. Moreover, there exist graph instances for which the cost of a shortest-path tree (or BFS tree) is at least $\Omega(n)$ times the cost of an optimal tree.*

Proof. The approximation factor follows from summing up equation (3.8) across all edges, and the fact that the shortest-path tree simultaneously minimizes both the costs in the left and right-hand side of the resulting inequality. For the lower bound, consider the graph shown in Figure 3.2. There are n triplets of nodes in parallel, node r as the root, and a final node labeled $3n + 1$. All nodes (except the root) have demand of $d_i = 1$, and all resistances are equal. On the left, we have the shortest-path tree (BFS tree) whose cost

[7] See the problem *SymT* in [67] for example.

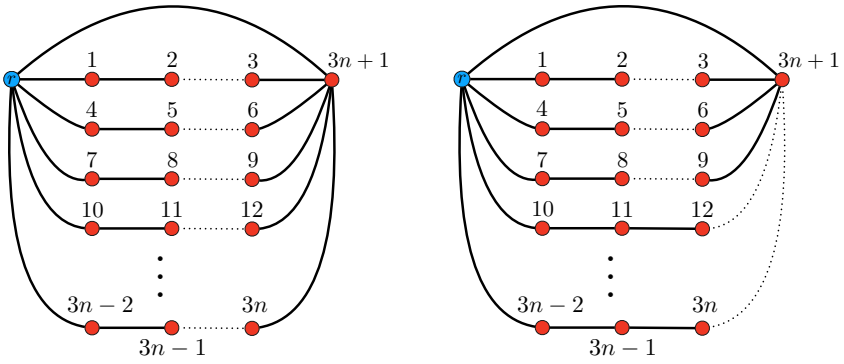

Figure 3.2: Lower-bound example on the performance of the shortest-path tree: (Left) shortest-path tree, versus (Right) optimal spanning tree.

can be calculated as:

$$APX = n \times (1^2 + 2^2) + (n+1)^2 + n \times 1^2 = n^2 + 8n + 1.$$

On the right, we have the optimal tree which does not change the first 3 triplets, but re-configures the rest as shown. The cost of this tree is

$$OPT = (n-3) \times (1^2 + 2^2 + 3^2) + 3 \times (1^2 + 2^2) + 4^2 + 3 \times 1^2 = 14n - 8.$$

Comparing the two costs proves a lower bound of $\Omega(n)$ on the performance of the shortest-path tree algorithm. □

3.5.2 Cut-based lower bounds

We now consider special graphs with a particular set of cuts that yield improved approximation factors. For the rest of this section we assume that all the edges of the graph have the same resistances, and without loss of generality we set $r_e = 1$ for all $e \in E$.

Theorem 3.5.2. *Consider a graph $G = (V, E)$ with root $r \in V$, and $r_e = 1, \forall e \in E$. Assume that G has a family of cuts $S_1, S_2, ..., S_\ell \subset V$ (with $r \in S_i, \forall i$) s.t.:*

(a) each edge $e \in E$ appears in at most M cuts, i.e., $|\{i : e \in \delta(S_i)\}| \leq M$ for all $e \in E$,

(b) the union of the cuts $\cup_i \delta(S_i)$ supports a spanning arborescence[8] A rooted at r such that (directed) edge $(u, v) \in A$ only if $u \in S_i, v \notin S_i$ for all $i \in [\ell]$ such that $e \in \delta(S_i)$.

Then, the approximation factor of the arborescence A is at most $M \times \max_i |\delta(S_i)|$.

Proof. Let $K = \max_i |\delta(S_i)|$ be the size of the biggest cut. We can use the second assumption of the theorem statement, to map any edge (u, v) of the arborescence to a cut S_i (if there is more than one cut, we can pick one arbitrarily). In this way, we split the cost of arborescence A among different cuts, while ensuring that the edges are carrying flows directed out of the cuts. Considering any of these cuts, we show that the costs of the trees A and OPT restricted to that cut are within a factor K, i.e., $\sum_{e \in \delta(S_i)} \mathcal{E}_A(e) \leq K \sum_{e \in \delta(S_i)} \mathcal{E}_{OPT}(e)$ for all i.

Let S be one of the cuts with $r \in S$, and let $k = |\delta(S)|$ be the size of the cut. Let $a_1, ..., a_k$ be the flow on the edges crossing the cut (calculated

[8] An r-arborescence is a directed spanning tree such that for any vertex v, there is exactly one directed path from r to v (see [68] for more details).

based on tree A), where $a_i > 0$ if the flow is going out of the cut and $a_i < 0$ if it is flowing inwards. With this choice of directions, the arborescence A has only non-negative a_i values. We also know that the total flow going across this cut is equal to the total demand below the cut, i.e., $\sum_{i=1}^{k} a_i = \sum_{v \notin S} d_v$.

Let $b_1, ..., b_k$ be the flow of the edges crossing this cut in the optimal tree, where some of these variables may be zero if the optimal tree is not using that edge, or even negative if they are being used in the opposite direction. However, we still know that $\sum_{i=1}^{k} b_i = \sum_{v \notin S} d_v$, and also the cost of this cut in the optimal tree is $\sum_{i=1}^{k} b_i^2$. This cost is minimized when all the b_i's are equal ($b_i = \frac{1}{k} \sum_{v \notin S} d_v$ gives a lower bound on the cost even if it is not attainable by any spanning tree.) Therefore we get:

$$\sum_{e \in \delta(S)} \mathcal{E}_{OPT}(e) = \sum_{i=1}^{k} b_i^2 \geq k \left(\frac{1}{k} \sum_{v \notin S} d_v \right)^2 \tag{3.9}$$

$$= \frac{1}{k} \left(\sum_{i=1}^{k} a_i \right)^2 \geq \frac{1}{K} \sum_{i=1}^{k} a_i^2 = \frac{1}{K} \sum_{e \in \delta(S)} \mathcal{E}_A(e), \tag{3.10}$$

where in the last inequality we dropped the (non-negative) cross-terms $a_i a_j$, and used the fact that the cut size $k \leq K$. If the cuts were disjoint, the K-approximation would directly extend to the entire objective function as well, as the cuts would divide the objective into separate additive objectives. However, we may double count what the optimal tree is paying since the cuts are not disjoint, but we know that each edge will be counted at most M times. This gives the approximation ratio of $M \times K$ in total. □

We use Theorem 3.5.2 for an $n \times m$ grid (where we let the number

of nodes be $N = mn$), and consider diagonal cuts as shown in Figure 3.3. Note that these cuts partition the edges of the grid, and their size is less than $2\sqrt{N}$. Any spanning tree that has edges only going to the right or the bottom, satisfies the second requirement of Theorem 3.5.2, and thus has cost at most $2\sqrt{N}$ times the optimal spanning tree.

Corollary 3.5.3. *There exists an $\mathcal{O}(\sqrt{N})$-approximation algorithm for minimizing the loss on an $n \times m$ grid with N nodes, when all the edges have the same resistance and the root is located at the corner of the grid.*

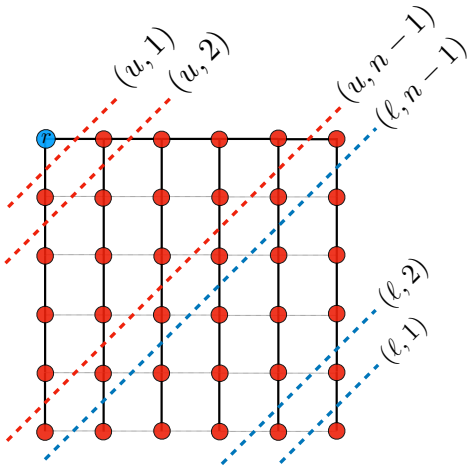

Figure 3.3: $n \times n$ grid with root at the top-left.

Constructing the above described set of cuts for general graphs remains an open question; planar graphs would be a natural candidate. By the planar separator theorem [69], we know that any N-node planar graph has a vertex separator of size $\mathcal{O}(\sqrt{N})$ that splits the graph into two (almost) equal parts. However, it is not clear how to find the desired family of cuts by using the

planar separator oracle. This is due to the second requirement of the cuts in Theorem 3.5.2 which fixes a natural direction on any edge once it appears in a cut. This direction should be respected in future cuts that include this edge; however, the separator oracle is oblivious to edge directions.

3.6 A 2-Approximation for Uniform Grid via Min-Min Algorithm

We now propose a constant-factor approximation algorithm for an $n \times m$ uniform grid ($n \leq m$) with the root at a corner of the grid (see Figure 3.3), in which all demands are equal ($d_i = 1$ for all $i \in V \backslash \{r\}$) and all resistances are equal ($r_e = 1$ for all $e \in E$). The key idea of considering this case is to help us better understand the structure of optimal solutions through combinatorial techniques, that can be generalized for real-life distribution networks (Figure 3.4). Even though the demands and resistances are uniform in the $n \times n$ grid, it is non-trivial to connect the loads together in a way that avoids big flow values close to the root. For example, Figure 3.3 demonstrates an example tree on an $n \times n$ grid, that satisfies the properties of both Theorem 3.5.1 and Theorem 3.5.2 solutions, and yet fails to provide a constant-factor approximation. The horizontal edges in this tree have flows of $n, 2n, ..., (n-1)n$, and therefore the cost of the tree is in the order of $\sum_{i=1}^{n-1}(i \times n)^2 = \mathcal{O}(n^5)$ (vertical edges have a total cost of $\mathcal{O}(n^4)$). On the other hand, the optimal cost in this grid is $\mathcal{O}(n^4 \log n)$, as we will show an $\Omega(n^4 \log n)$ lower bound in Lemma 3.6.1 and prove a constant-factor approximation in the rest of this section. For the sake

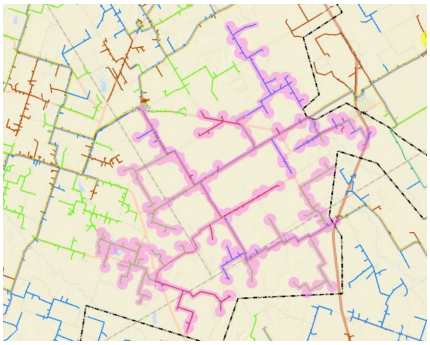

Figure 3.4: Snapshot of an anonymized real distribution network from a utility company in the US. These networks typically look like sparse subgraphs of grids, thus motivating our exploration of approximations in Section 3.6 and computational experiments in Section 3.8.

of brevity, we will present our novel MIN-MIN algorithm for a square $n \times n$ grid, while the results hold for rectangular grids as well and more general demands.

Notation: We let $n \times n$ be the size of the grid and the total number of nodes be $N = n^2$. We consider the diagonal cuts as shown in Figure 3.3 and we name them (u, i), $i = 1, ..., n-1$ for the upper triangle, and (ℓ, i) for the lower triangle. Note that the diagonal cuts cover all the edges and each edge appears in only one cut. Therefore, we can divide the cost of any spanning tree (either optimal or approximate tree) into the costs from each cut. Let $OPT^{(u,i)}$ and $Alg^{(u,i)}$ denote the cost of edges that cross cut (u, i) in the optimal and approximate solution, respectively. Similarly we define $OPT^{(\ell,i)}$ and $Alg^{(\ell,i)}$ for the lower triangle. Finally, let $OPT^u = \sum_{i=1}^{n-1} OPT^{(u,i)}$ and $OPT^\ell = \sum_{i=1}^{n-1} OPT^{(\ell,i)}$. Then, we have $OPT = OPT^u + OPT^\ell$. Similarly, we define Alg^u and Alg^ℓ.

In Section 3.6.1, we will first use these diagonal cuts to find a lower bound for the cost of any spanning tree. In Section 3.6.2, we explain our Min-Min algorithm that will crucially use these diagonal cuts, and give its performance guarantee in Section 3.6.3.

3.6.1 Lower bounds

Let $S_{u,i}$ and $S_{\ell,i}$ be the number of nodes below cuts (u,i) and (ℓ,i), respectively. Then, $S_{u,i} = n^2 - \sum_{j=1}^{i} j = n^2 - \frac{i(i+1)}{2}$, and $S_{\ell,i} = \sum_{j=1}^{i} j = \frac{i(i+1)}{2}$.

Upper triangle cuts: By a quick look at the structure of the grid, we can observe that there are $2i$ edges that cross the cut (u,i). These edges are connected to i nodes on the root side of the cut, and $i+1$ nodes on the other side. Call these nodes $u_1, ..., u_i$ on the root side and $v_1, ..., v_{i+1}$ below the cut. We call an edge (u_j, v_k) of the tree *outgoing*, if u_j is the parent of v_k in the tree. We claim that the tree can have at most $i+1$ outgoing edges over this cut, although it can have all the $2i$ edges. This is because if we have more than $i+1$ outgoing edges, then by the pigeonhole principle, a node v_k will have two parents and this creates a loop in the tree.

In addition, all the $S_{u,i}$ nodes below the cut are connected to the root through (at least) one of these outgoing edges over this cut. This is true because we can traverse the path from the root to that node, and at some point we must cross the cut through an outgoing edge. It is possible to have multiple outgoing edges on that path if we cross the same cut multiple times,

but all we need is that each node below the cut is counted as a successor for at least one of the outgoing edges. So the aggregate number of successors for the outgoing edges of cut (u, i) is at least $S_{u,i}$, while there are at most $i + 1$ such edges. Recall that when the summation of a number of variables is fixed, their sum of squares is minimized when all of them are equal. Hence, in the most balanced way any tree (including the optimal tree) has to pay the following cost over this cut:

$$OPT^{(u,i)} \geq (i+1) \times \left(\frac{S_{u,i}}{i+1}\right)^2 = \frac{S_{u,i}^2}{i+1}. \tag{3.11}$$

By summing the above lower bound over different cuts, we can also get the following lower bound on the energy of any spanning tree over the entire upper triangle.

Lemma 3.6.1. *The cost of the optimal tree over the upper triangle part of the grid is lower bounded by:* $OPT^u \geq \Omega(n^4 \ln(n))$.

Proof. To obtain a lower bound for OPT^u, we just plug the value of $S_{u,i}$ in (3.11) and sum over i:

$$\begin{aligned}
OPT^u &\geq \sum_{i=1}^{n-1} \frac{S_{u,i}^2}{i+1} = \sum_{i=1}^{n-1} \frac{\left(n^2 - i(i+1)/2\right)^2}{i+1} \\
&= \sum_{i=1}^{n-1} \frac{n^4}{i+1} + \frac{1}{4}\sum_{i=1}^{n-1} i^2(i+1) - \sum_{i=1}^{n-1} n^2 i \\
&\geq n^4\big(\ln(n+1) - 1\big) + \frac{(n-1)^2 n^2}{16} + \frac{(n-1)n(2n-1)}{24} - \frac{n^3(n-1)}{2} \\
&= n^4 \ln(n+1) - \frac{23}{16}n^4 + \frac{11}{24}n^3 - \frac{1}{16}n^2 + \frac{1}{24}n.
\end{aligned}$$

□

Lower triangle cuts: The argument here is exactly the same as in the upper triangle except that there are $i+1$ nodes on the root side and i nodes on the other side. Therefore, in the most balanced case when all the outgoing edges carry the same flow, we have i outgoing edges with flows $S_{\ell,i}/i = (i+1)/2$, paying the total cost of:

$$OPT^{(\ell,i)} \geq i \times \left(\frac{S_{\ell,i}}{i}\right)^2 = \frac{i(i+1)^2}{4}. \tag{3.12}$$

3.6.2 Min-Min algorithm

The Min-Min algorithm builds a spanning tree which contains n disjoint paths with different lengths over the lower triangle; see the blue paths in Figure 3.6 (right). Then, in each cut of the upper triangle, exactly one pair of subtrees merge together. As the name suggests, we merge the two subtrees with the minimum number of successors in each step, and call it the *merging step*. However, this requires those two subtrees to be next to each other. Therefore, we need to order our disjoint paths in the lower triangle in a way that allows merging minimum load subtrees in the upper triangle; we call this the *uncrossing step*. In the following lemma, we show that the number of successors of edges on the main diagonal can be any permutation of the numbers $1, 2, ..., n$.

Lemma 3.6.2 (Disjoint paths)**.** *We can obtain a disjoint path decomposition of the lower triangle of the grid (i.e., a set of n paths from the diagonal that are shortest between the end-points and cover all the vertices in the lower triangle)*

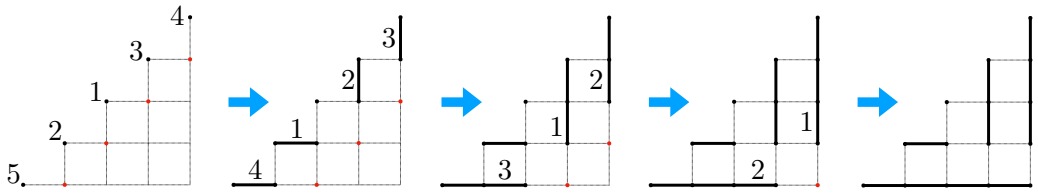

Figure 3.5: Example of Lemma 3.6.2. In each step we decrease the number of successors desired by one and put them on the next diagonal (red nodes), except 1 which is already satisfied.

for any ordering of numbers $1, 2, ..., n$*, specifying the number of successors of edges on the left diagonal of the grid.*

Proof. We give a recursive construction which also proves the existence of such paths. Let $(a_1, a_2, ..., a_n)$ be a permutation of $(1, 2, ..., n)$. Put these numbers on the main diagonal. Except $a_i = 1$ which is already satisfied, connect the rest of the nodes to the nodes of the next diagonal, which has $n - 1$ nodes, in the same order. Now we have to construct a permutation of $(1, 2, ..., n - 1)$ on this new diagonal, because the previous numbers should be decreased by one. We can repeat the process. An example is performed in Figure 3.5. □

Note that Lemma 3.6.2 ensures the adjacency of minimum subtrees in all upper triangle cuts. To get the right permutation, we can start from any permutation (say $1, 2, ..., n$) and do the Min-Min merging as shown in Figure 3.6 (left). Then we can do the uncrossing from top to bottom as shown in Figure 3.6 (middle) and this gives the desired permutation on the main diagonal. Finally, we can construct the lower part of the spanning tree corresponding to that permutation using Lemma 3.6.2, and construct the upper

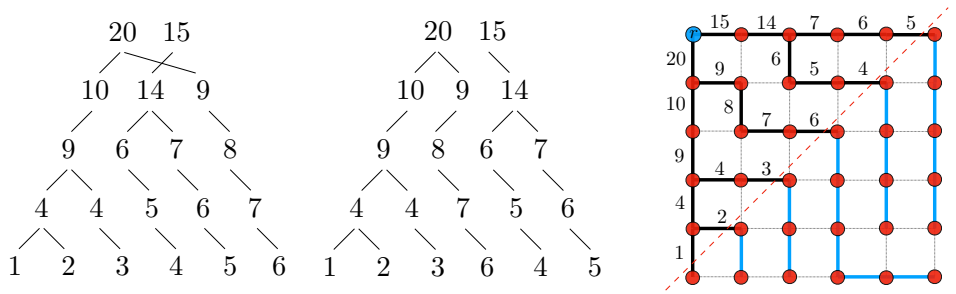

Figure 3.6: Example of the MIN-MIN algorithm. **Left**: Merging the two smallest numbers in each layer, starting from the $1, 2, ..., n$ sequence. **Middle**: Same tree re-ordered from top to bottom to avoid crossings. **Right**: The corresponding grid where the lower triangle is constructed by Lemma 3.6.2, and the numbers in the upper triangle are merged in each diagonal layer according to the middle tree.

part of the spanning tree by merging minimum size subtrees in each layer. An example of the algorithm is shown in Figure 3.6 (right). The formal description of the algorithm for general rectangular grids is also tabulated under Algorithm 3, in which for the case of a rectangular grid, we use the middle part of the grid to connect the lower triangle to the upper triangle via parallel disjoint paths.

3.6.3 Approximation factor for the MIN-MIN algorithm

Lower triangle: We first show that the cost of MIN-MIN algorithm is at most 4/3 of the optimal tree, over the lower triangle.

Lemma 3.6.3. *The* MIN-MIN *algorithm costs at most* 4/3 *of the optimal over the cuts in the lower triangle. In other words,*

$$Alg^{(\ell,i)} \leq \frac{4}{3} OPT^{(\ell,i)}, \quad i = 1, 2, \ldots, n-1,$$

Algorithm 3 MIN-MIN algorithm

Input: An $n \times m$ grid ($n \leq m$).

1: Start with sequence $(m-n)+1, (m-n)+2, ..., (m-n)+n$.
2: **while** there are more than two numbers **do**
3: Merge the two smallest numbers.
4: Add one to the entire sequence.
5: **end while**
6: **Uncrossing**: Backtracking the previous step, find the right permutation of the starting sequence to put on the diagonal cut $(u, n-1)$, to ensure the adjacency of smallest two numbers in all steps.
7: **Parallel paths**: Use parallel paths of length $m-n$ to connect the nodes below cut $(u, n-1)$ to the nodes above cut $(\ell, n-1)$.
8: **Disjoint paths**: Subtract $m-n$ from the sequence of Step 6. Now use Lemma 3.6.2 on this sequence to form the disjoint paths on the lower triangle.
9: **Merging**: Complete the spanning tree by merging the two smallest subtrees in every upper triangle cut.

where Alg refers to the output of MIN-MIN *algorithm. Moreover, this implies the same approximation factor for the entire lower triangle energy, i.e.,* $Alg^{\ell} \leq \frac{4}{3} OPT^{\ell}$.

Proof. By the construction of Lemma 3.6.2, the edges of the proposed tree over cut (ℓ, i) have $1, 2, ..., i$ successors (in some order). Therefore,

$$Alg^{(\ell,i)} = \sum_{j=1}^{i} j^2 = \frac{i(i+1)(2i+1)}{6}, \quad i = 1, \ldots, n-1.$$

Comparing to the lower bound (3.12) for lower triangle cuts:

$$\frac{Alg^{(\ell,i)}}{OPT^{(\ell,i)}} \leq \frac{4i(i+1)(2i+1)}{6i(i+1)^2} = \frac{2(2i+1)}{3(i+1)} < \frac{4}{3}.$$

Since this is true for all cuts, it also holds for the entire lower triangle. □

Upper triangle: To analyze the MIN-MIN algorithm over the upper triangle, we first obtain the following relation between the cost of the algorithm over different cuts.

Lemma 3.6.4. *For $i = 1, \ldots, n-2$, we have*

$$Alg^{(u,i)} \leq Alg^{(u,n-1)} + 2\sum_{j=i+1}^{n-1}\left[S_{u,j} + \frac{S_{u,j}^2}{j(j+1)} + j\right]. \tag{3.13}$$

Proof. Let $a_1, a_2, ..., a_{i+1}$ be the number of successors for the edges of cut (u, i), in non-decreasing order ($i \geq 2$). We know that $\sum_{j=1}^{i+1} a_j = S_{u,i}$. Since we merge a_1, a_2 in the higher level, the edges of cut $(u, i-1)$ will have $(a_1+a_2+1), (a_3+1), ..., (a_{i+1}+1)$ successors. Therefore the cost of cut $(u, i-1)$ is

$$\begin{aligned} Alg^{(u,i-1)} &= (a_1 + a_2 + 1)^2 + (a_3 + 1)^2 + ... + (a_{i+1} + 1)^2 \\ &= \sum_{j=1}^{i+1} a_j^2 + 2\sum_{j=1}^{i+1} a_j + 2a_1a_2 + i = Alg^{(u,i)} + 2S_{u,i} + 2a_1a_2 + i. \end{aligned}$$

Since a_1 is the smallest number, it is upper-bounded by the average, i.e. $a_1 \leq S_{u,i}/(i+1)$. Similarly, a_2 is the smallest among the rest, therefore $a_2 \leq (S_{u,i} - a_1)/i \leq S_{u,i}/i$. So the algorithm satisfies:

$$Alg^{(u,i-1)} \leq Alg^{(u,i)} + 2S_{u,i} + 2\frac{S_{u,i}^2}{(i+1)i} + i.$$

By recursively applying this upper bound, we get (3.13). □

Next, add equation (3.13) across all upper-triangle cuts and use the lower bound of Lemma 3.6.1 to upper bound the energy of Alg over the upper triangle:

Lemma 3.6.5. *The output of the* Min-Min *satisfies:* $\frac{Alg^u}{OPT^u} \leq 2 + \mathcal{O}(\frac{1}{\log n})$.

Proof. We first sum (3.13) over i to get:

$$Alg^u = \sum_{i=1}^{n-1} Alg^{(u,i)} \tag{3.14}$$

$$\leq (n-1)Alg^{(u,n-1)} + 2\sum_{i=1}^{n-2}\sum_{j=i+1}^{n-1}\left[S_{u,j} + \frac{S_{u,j}^2}{j(j+1)} + j\right] \tag{3.15}$$

$$= (n-1)Alg^{(u,n-1)} + 2\sum_{j=2}^{n-1}(j-1)\left[S_{u,j} + \frac{S_{u,j}^2}{j(j+1)} + j\right] \tag{3.16}$$

Now, for the first term in (3.16) we have:

$$(n-1)Alg^{(u,n-1)} = (n-1)\sum_{i=1}^{n} i^2 = \frac{(n-1)n(n+1)(2n+1)}{6}.$$

For the $S_{u,j}$ term in the summation of (3.16) we have:

$$2\sum_{j=2}^{n-1}(j-1)S_{u,j} = 2\sum_{j=2}^{n-1}(j-1)\left(n^2 - \frac{j(j+1)}{2}\right) = \frac{3}{4}n^4 - \frac{5}{2}n^3 + \frac{9}{4}n^2 - \frac{1}{2}n.$$

For the quadratic term in (3.16), we have:

$$2\sum_{j=2}^{n-1}\frac{(j-1)S_{u,j}^2}{j(j+1)} \leq 2\sum_{j=2}^{n-1}\frac{S_{u,j}^2}{j+1} \leq 2\sum_{j=2}^{n-1}OPT^{(u,j)} \leq 2OPT^u,$$

where the second inequality is due to (3.11). Finally, we can calculate the last term of (3.16), as:

$$2\sum_{j=2}^{n-1}(j-1)j = \frac{2}{3}n^3 - 2n^2 + \frac{4}{3}n.$$

Replacing these polynomials back into (3.16) we get:

$$Alg^u \leq 2OPT^u + \frac{13}{12}n^4 - \frac{5}{3}n^3 - \frac{1}{12}n^2 + \frac{2}{3}n. \tag{3.17}$$

Dividing this by OPT^u and using the lower bound of Lemma 3.6.1 completes the proof. □

Overall approximation: Once we have separate approximation guarantees for OPT^ℓ and OPT^u, the worse approximation factor determines the overall result.

Theorem 3.6.6. *For a rectangular $n \times m$ ($n \leq m$) grid with loads satisfying $d_i \in [d_{\min}, d_{\max}]$ for all nodes $i \in V\backslash\{r\}$, the* Min-Min *algorithm for the network reconfiguration problem with uniform resistances gives an approximation factor of $\alpha^2\left(2 + \mathcal{O}(\frac{1}{\log n})\right)$, where $\alpha = d_{\max}/d_{\min}$. In particular, if the loads are uniform and as $n \to \infty$, the* Min-Min *algorithm gives a 2-approximation.*

Proof. For square grids with uniform loads, the approximation result follows immediately from Lemmas 3.6.3 and 3.6.5. Moreover, for the rectangular grid with uniform loads, we can apply the Min-Min algorithm on the lower and upper triangle parts, and use the middle section to connect the two triangles simply by parallel disjoint paths. Then the analysis would go through exactly as the square case.

For non-uniform loads, we consider the uniform counterpart of this instance, which is the same graph with $d_i = d_{\min}$ for all $i \in V\backslash\{r\}$. Running the Min-Min algorithm on this uniform case outputs a tree T whose loss is at most twice of the optimal tree in the uniform setting (call the optimal tree T_u^*, and its loss OPT_u). Let f and $\tilde{f}$ be the electrical flows on tree T with the

actual and modified loads, respectively; then we have $\tilde{f} \leq f \leq \alpha\tilde{f}$. This gives the following inequality regarding energies:

$$\mathcal{E}(f) \leq \alpha^2\mathcal{E}(\tilde{f}) \leq \alpha^2\big(2 + \mathcal{O}(\frac{1}{\log n})\big)OPT_u$$

It only remains to argue that $OPT_u \leq OPT$, where OPT is the loss of the optimal tree (call it T^*) in the original instance. This is true because if we reduce the loads on T^* to $d_{\min}$, we decrease its loss, but on the other hand, the resulting energy should still be more than OPT_u, by the optimality assumption of OPT_u.

Note that α in the above approximation captures the ratio between the biggest and the smallest loads, and is usually independent of n in practice (typically, loads do not vary a lot in a realistic scenario). However, for the sake of completeness, the above approximation ratio can be thought of as $\min\{2\alpha^2, n\}$, where n comes from the general result of Theorem 3.5.1 in the case of a big α. □

3.7 A Generalization of Min-Min: Layered-Matching

To make our results applicable to general settings, we extend our intuition from the theoretical results and propose a generalization of Min-Min algorithm. The main idea is to partition the graph into layers and connect each layer to the upper layer in a balanced way, similar to Min-Min. To partition the graph into layers, we can use an arbitrary breadth first search tree rooted at r, which results in layers based on the hop-distance from the root (this is

like the diagonal cuts on the grid). To connect layers in a balanced manner, we use the solution of the flow relaxation, and find the best matching that creates a flow which is close to the relaxed solution in terms of L_∞ norm. In particular, let us assume that we aim to connect nodes in layer L_k to the upper layer L_{k-1}. Let f^r be the solution of the flow relaxation, d_i be the demand of node i, and x_{ij} be the indicator of picking an edge $(i,j) \in E$. Then, finding the best matching reduces to solving the following integer program:

$$\begin{aligned} &\min \epsilon \\ &\text{s.t} \sum_{\substack{i \in L_{k-1}: \\ (i,j) \in E}} x_{ij} = 1 && \forall\, j \in L_k \\ &\quad -\epsilon \leq x_{ij} d_j - f^r_{ij} x_{ij} \leq \epsilon && \forall\, i \in L_{k-1}, j \in L_k : (i,j) \in E \\ &\quad x_{ij} \in \{0,1\} && \forall\, i \in L_{k-1}, j \in L_k : (i,j) \in E \end{aligned} \tag{3.18}$$

This IP can be solved very fast in practice since it finds a (local) matching between two layers of nodes (as discussed in the next section). Once we find the matching between layers k and $k-1$, we contract each node of L_{k-1} with its successors, while replacing the demand of that node with the total demand of the corresponding subgraph. By repeating this process in a bottom-up fashion, all the nodes get connected to the root via a single path, hence we obtain a valid spanning tree. The full description of the heuristic is in Algorithm 4.

3.8 Simulation Results

As previously discussed, electricity distribution networks resemble sparsified grids. Hence, for our computational experiments, we constructed 25 in-

Algorithm 4 Layered-Matching Heuristic

1: Compute BFS tree rooted at the root r.
2: Let $L_0, \ldots, L_t$ be the layers formed in the BFS tree, where $L_0 = \{r\}$.
3: **for** $k = t, t-1, ..., 1$ **do**
4: Match each node of L_k to exactly one node in L_{k-1} by solving (3.18).
5: **for** $j \in L_{k-1}$ **do**
6: Contract subgraph rooted at j into a single node with demand equal to that of the entire subgraph.
7: **end for**
8: **end for**

stances on 25×25 grids where we sample edges for deletion independently with some probability $p \in \{0.05, 0.1, 0.2\}$ and delete that edge only if it does not disconnect the graph; an example of such sparsified grids is given in Figure 3.7. Moreover, we consider demands randomly chosen in the interval $[0.5, 1.5]$, resistances randomly chosen in the interval $[1, 10]$ and the root in the corner. To incorporate the acyclic support constraint, we utilized Martin's [70] extended formulation for spanning trees.

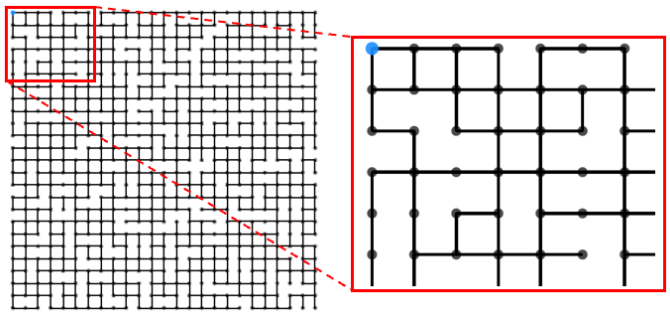

Figure 3.7: An abstraction of a real-life distribution network through an instance of a sparsified 25×25 grid that was used in the computations, where the sparsification probability is $p = 0.2$ and the root is in the top left corner.

We benchmarked[9] the performance of depth-first search (DFS) trees, RIDE, the Layered-Matching (LM) heuristic, the Branch Exchange heuristic and the convex integer program. The Branch Exchange variant we utilized starts a new iteration once an improving solution is found, as opposed to looking for the exchange that results in the most improvement. We also considered a variant that uses binary search on the value of the improvement where we only do an exchange if it results in an improvement of at least T; if no such improvement exists we divide T by two and proceed. We found that the latter version was significantly slower.

Our computations show that the algorithms proposed in this paper are orders of magnitude faster than branch exchange (initialized with a random DFS tree) while having comparable performance. For example, on 25×25 grid instances with sparsification probability $p = 0.2$, the mean time taken by the LM heuristic is 1.35 seconds, whereas the mean time it takes the Branch Exchange heuristic to attain the same cost as LM is around 35,000 seconds (see Table 3.1). We believe that this improvement is significant and will allow system operators to minimize losses even on an hourly basis as demand patterns change.

Moreover we find that `Gurobi` mostly failed to even find a feasible solution for sparsified 25×25 grids instances in a 24-hour time limit. To help the IP solver, we provide a warm start solution using different algorithms and let

[9]We implemented all algorithms in `Python` 3.7, and used `Gurobi` 9 [71] as a solver for the MIP.

Table 3.1: Comparing the mean optimality gap from best feasible solution found ($\times 100\%$) and mean running time (secs) of different algorithms on sparse 25×25 grids with varying probability of sparsification (p).

p	DFS		RIDE		LM		Branch Exchange time
	Gap	Time	Gap	Time	Gap	Time	to attain gap of LM
0.05	60.80	0.57	8.13	196.82	1.22	**2.93**	**5680.30**
0.1	49.69	0.35	6.36	125.89	1.12	**1.92**	**9380.28**
0.2	39.43	0.21	5.73	77.07	0.90	**1.35**	**35181.11**

the solver run for 24 hours. This outperforms the Branch Exchange heuristic after running for the same one day time limit (see Figure 3.8). In addition, running the MIP with the LM output as a warm-start obtains the same performance as initializing Branch Exchange with the LM output (as opposed to a DFS tree), however the MIP additionally gives provable optimality gaps. We report the gap in solution quality with respect to the best feasible solution found after running all algorithms for 24 hours. In particular, we found that the best feasible solution was always obtained by either running the MIP with the LM output as a warm-start or Branch Exchange initialized with the LM output.

Finally, the computations suggest that the approximation factor obtained by RIDE in practice is in fact much better than the worst-case theoretical bounds we showed. Furthermore, the performance of RIDE (as expected) and LM significantly improve as the graph gets sparser (see Table 3.1), and their improvement over Branch Exchange becomes more pronounced. This is very desirable since electricity distribution networks are indeed typically

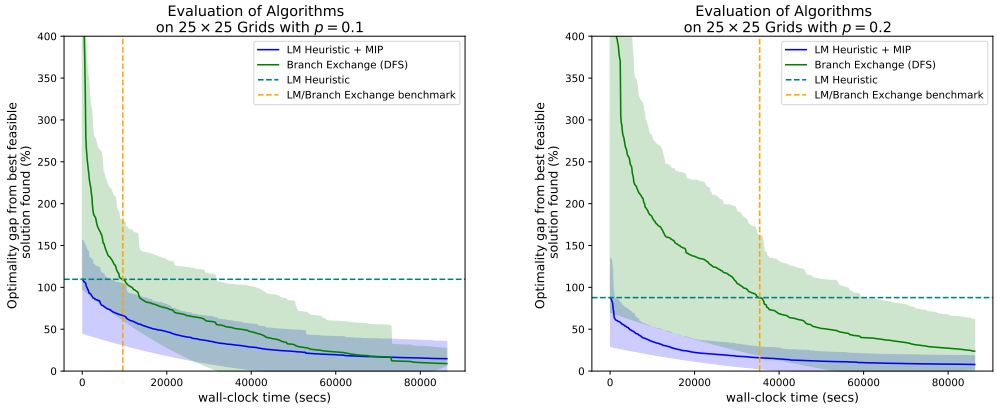

Figure 3.8: Plots comparing performance of branch exchange (initialized with DFS trees) and the mixed integer program initialized with the layered matching heuristic solution, on sparse 25×25 grids with sparsification probability $p = 0.1$ (left) and $p = 0.2$ (right). The demands and resistances are randomly chosen in $[0.5, 1.5]$ and $[1, 10]$ respectively. The dotted horizontal line compares the quality of the Layered Matching heuristic solution and the color margins represent confidence intervals across the different instances.

sparse in practice.

3.9 Conclusion

In this chapter we studied the network reconfiguration problem (for electricity distribution networks) through the lens of approximation algorithms. We provided approximation algorithms for different scenarios with restrictions on graph structure, line resistances, and node demands. A large number of open questions still remain, including the extension of the $\sqrt{n}$-approximation (or even constant-factor approximation) to planar graphs, analysis for the iterative deletion of the min-flow edge (introduced by Shirmohammadi and Hong [32]), and the hardness of the problem for grids or planar graphs.

Chapter 4

Algorithms with Guarantees for Mobile Energy Storage Dispatch

This chapter presents the development and implementation of approximation algorithms to optimally dispatch mobile storage for power grid support. The two main themes are algorithm development and the storage dispatch concept, with the latter detailing both the utility and electric vehicle (EV) owner perspectives. The upward trend in EV growth, coupled with the variability of solar and wind penetration, are increasingly straining the power grid. With this, we ask, what if we could redirect EV usage to balance rather than further strain the grid?

In our storage dispatch concept, the utility sets up an agreement offering time and location varying rewards to a central planner for the dispatch of storage to utility designated locations of need. The planner then computes an optimal dispatch of which batteries to utilize where and when to maximize its collected rewards, based on the varying battery availabilities.

Our contributions are A) the theoretical development for state-of-the-art dispatch of a varying set of batteries at multiple time, location and price-varying distribution nodes, and B) a description of our concept with its poten-

tial real-world impact, theory-to-concept and other implementation considerations. Our theoretical results include a new strong NP-hardness result, as well as three approximation algorithms with worst-case performance guarantees of $1/3$, $1/2$ and $1 - 1/e \approx 0.63$ and empirical performance between 82%-100%.[1]

4.1 Introduction

With the global COVID-19 pandemic, the world is looking more and more into speeding up the progression of transition to Net-Zero emissions [73]. For the first time in history, oil producers have been forced to sign zero emissions agreements, meaning they have to offset their oil production with green alternatives. Further, it's largely agreed that oil demand is likely past its all-time historic high and will only continue to decline going forward. Leading this decline is the electrification of transportation, which has surged this year with electric vehicle technology innovations, including Tesla's million-mile battery [74], the new electric/hydrogen trucks from Nikola [75] and expanding ride-sharing EV fleets by Tesla's Robotaxis and Facedrive. Between this upward trend in EV growth and the variability of renewable penetration increasingly straining the power grid, we ask, *what if we could redirect EV usage to balance rather than further strain the grid?*

As an example of a strained grid, consider the image of a real-world

[1]This chapter is based on the manuscript submitted to the IEEE Transactions on Power Systems [72]. The author of this dissertation made the key technical contributions to this work.

Figure 4.1: Thermal violations on a real distribution feeder.

distribution feeder shown in Figure 4.1. Only twenty hours of thermal violations over an entire year force a five-mile line upgrade at a cost to the utility of $400,000. These violations occurred in 2018 and the subsequent required upgrade was completed by the Texas-based utility in 2019. Had these twenty hours of violations been mitigated using alternative methods, such as EV batteries, a single battery discharge of say 75 kWh (an average EV battery size) would have been worth up to $225 to the utility.[2]

The above situation is typical of utility feeders facing required upgrades, usually caused by load growth, new EV charging demand and added variabil-

[2]Amortized over 20 years including interest, this upgrade effectively costs the utility $30,000 per year. If our mobile storage solution were used for these twenty hours of violations, rather than the line upgrade, it would have resulted in an opportunity cost of $1,500 for each of these 20 critical hour. The twenty hours of thermal violations averaged 500kWh per critical hour (approximately 10% over line rating) with a cumulative 10MWh total. Utilizing several EVs with battery capacities of 75 kWh, discharged during each critical hour, could have prevented the thermal overload and resulting line upgrade. Thus each EV discharge of 75 kWh could have been worth up to $225 to the utility during these critical hours. As an aside, this $225 equates to $3/kWh. Compared to $0.10/kWh, this implies a cost to the utility that is 30 times higher than their typical customer electricity rate.

ity from residential solar. This particular utility has around 60 feeders (other larger utilities may have hundreds of feeders) that have many cost saving opportunities—thermal, voltage, demand charges and others thus justifying there is a market for flexible storage. It also demonstrates the need for efficient dispatch algorithms to meet the scale of thousands of potential benefit nodes in each network, paired with thousands of participating EVs. With the world heading towards higher energy efficiency and increasing residential solar and plug-in EVs adding more disturbances to the grid, flexible storage could provide viable short-term and long-term solutions for many distribution feeders.

Motivated by the above, we envision our mobile storage solution that uses EVs to mitigate both critical and non-critical violations. Our proposed solution can be much more versatile and cost-effective than traditional solutions, which are typically limited to line/equipment upgrades, bulk load shedding and demand response. Specifically, our solution can be effective in mitigating thermal, voltage and other violations, reducing transmission demand peak charges, improving short-term voltage control, and increasing system efficiency (through reduced losses, preservation of equipment life and improved three-phase balancing).

In our proposed solution, the utility needs to announce rewards at different times and locations of need, capturing the above-listed objectives, and then a central planner calculates a dispatch each day for moving her available batteries to these points of need.

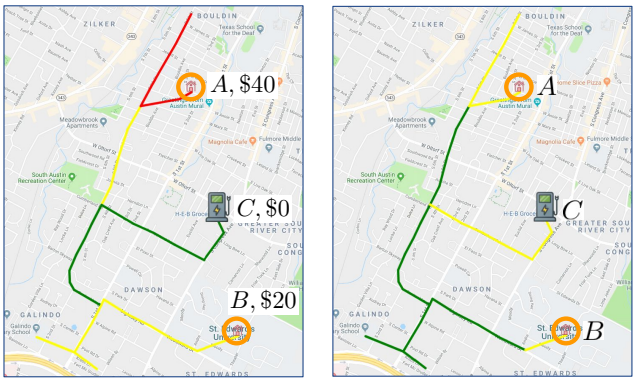

Figure 4.2: Example of eliminating violations via our mobile storage solution.

To illustrate our solution's potential benefit to the grid, Fig. 4.2-(left) shows an example where, in the absence of a mobile storage solution, feeders A, B and C have: critical thermal or other violations (red), other non-critical cost-saving opportunities (yellow) and a good operating condition (green), respectively. The prior day, the utility announces rewards of \$40, \$20 and \$0 for, say, 2 pm today. For calculation purposes, \$40 is derived from a 75kWh battery. (Compare to the \$225 EV discharge value in the real-world example above.) In Fig. 4.2-(right), batteries are discharged and collect the rewards at points A and B (both at schools) and charged at point C (a grocery store), and consequently the violations are eliminated.

In this new age of increasing renewable and EV penetration, utilities face the real need to become innovative and our solution can be a valuable and cost-effective new tool for their toolboxes. Stepping into the near future, we also see our solution as complementary to Tesla's 2020 *robotaxis* launch [76], with Tesla's CEO citing fleet re-charging as an open challenge. This chapter's

technical contribution is directly relevant by proposing algorithms for optimal dispatch of charging and discharging of EVs, and our solution could add another dimension to ride-sharing business models, where EVs can be profitably used in conjunction for both ride-sharing and grid support.

The main contributions of this chapter are the following:

1. We propose a mobile storage solution designed to: (1) mitigate thermal and voltage violations; (2) reduce transmission and other demand charges; (3) improve short-term voltage control; and (4) increase system efficiency through reduced losses, extended equipment life and improved three-phase balancing. Our solution computes a dispatch each day for moving available batteries to points of need, as determined by the utility via a set of rewards reflecting its above objectives.

2. To optimize the battery discharge dispatch, we set up a mathematical model and prove that the problem is strongly NP-hard.

3. Despite the computational hardness of the general setting, we obtain the optimal solution for several special cases of the problem, when (i) we have access to super-fast chargers, (ii) we have a single battery, (iii) the number of batteries is constant, or (iv) all batteries have constant charging time and the same availability. We use these special cases as insights to the theoretical nature of the problem and building blocks for solving the general setting next.

4. We then propose three approximation algorithms, GREEDY, GREEDY+, and RANDOMIZED ROUNDING, and we prove that they provide (worst-case) performance guarantees of 1/3, 1/2, and $1 - 1/e \approx 0.63$, respectively. We also consider a fourth algorithm called BOOSTED RANDOMIZED ROUNDING.

5. We test the empirical performance of the four algorithms, and show that they perform much better than their worst-case theoretical guarantees, as they achieve at least 82% of the optimal reward in all test cases. Somewhat surprisingly, the best empirical performer is GREEDY+ with greater than 95% performance, even though RANDOMIZED ROUNDING has the best theoretical guarantee.

4.2 Related Work

Energy storage systems are now an inevitable part of the electricity grid, due to the penetration of intermittent renewable energy sources. Even though stationary storage systems have been extensively studied in the literature [77–80], there is much less work on mobile storage systems [81], especially on mobile storage dispatch, namely where and when these mobile storage units should be charged and discharged. Behind-the-meter battery storage has also been considered for eliminating congestion and voltage violations [82–84].

Most research on mobile storage systems focuses on electric vehicle use, which can provide various services to the grid. EV charging and discharging

has previously been studied in the context of power loss minimization [85, 86], social welfare maximization [87], frequency regulation [88], voltage regulation [85, 89], peak shaving [90–92], valley filling [93], and supporting renewable energy sources [90, 94]. We refer the reader to a recent survey by Amjad *et al.* [95] on various optimization approaches and objectives employed for EV charging.

Related to our mobile storage concept, He *et al.* [96] and Hutson *et al.* [97] consider charging and discharging of EVs together with EV availabilities. In contrast to our work, neither of these two papers considers different prices at different nodes. The mobility of EVs is utilized by Timpner and Wolf [98], but in a very different scenario, as they only consider the movements within a parking lot so as to share a limited number of charging ports in a single location.

Another related result is that of Abdeltawab and Mohamed [99], in which the authors aim to maximize the profit of distribution network operators by optimizing the day-ahead schedule of a (single) storage truck. Our work differs in that we consider many storage devices across the network versus only one. Also, in contrast to this chapter, they do not provide any optimality guarantee as they utilize the heuristic approach of particle swarm optimization to find a profitable dispatch. Furthermore, as we show in this chapter, it is the multiplicity of storage units that makes the problem computationally hard.

Recently, Krishna *et al.* [100] proposed an EV charging ecosystem which is in a sense the reverse of our concept. In their model, the central planner is

responsible for matching the EV users to private garage chargers (which are typically under-utilized) in order to alleviate the demand-supply mismatch in public charging. An online competitive algorithm for a similar problem was proposed by Sun *et al.* [101], where the charging network operator aims to maximize its expected total revenue via offering different charging station and prices to EV owners.

Finally, the dispatch of mobile power sources, such as truck-mounted emergency generators, has been studied in the case of natural disasters [102–104]. However, the objective is very different in those extreme situations as the goal is to restore the power of critical loads as soon as possible, in other words to minimize the interruption cost.

From an algorithmic perspective, most existing work is either based on mixed-integer programs [82, 84, 105], which cannot be solved efficiently for large instances, or heuristics without optimality guarantees. These heuristics include genetic algorithms [106], particle swarm optimization [97, 107, 108], and ant colony optimization [109]. In contrast, we exploit techniques from theoretical computer science to provide novel hardness results and approximation algorithms, which are scalable algorithms with rigorous performance guarantees.

We see our work belonging to the small but growing body of research bridging theoretical computer science and power systems, specifically utilizing tools from approximation algorithms to provide rigorous guarantees for the mobile storage dispatch problem. Other related work employing rigorous

theoretical analysis for power systems problems includes: repair and restoration in power systems [110, 111], complexity of switching in transmission and distribution systems [5, 48, 50], energy plan selection in retail markets [112], energy sharing of prosumers [113, 114], energy-efficient truck transportation [115], and voltage control in smart grids [116].

4.3 Mobile storage solution: Concept

Utility solution set-up: Our mobile storage solution is to connect batteries to the distribution network at locations and times of utility planning and operational need. Our solution takes as input distribution nodal costs provided by the utility, which may incorporate: (1) thermal, voltage and other violations; (2) transmission demand charges and seasonal peak-related charges[3]; (3) short-term voltage control; and (4) system efficiency (losses, equipment life and three-phase balancing).

Most utilities already do periodic historical power flow analysis every 1-4 years to determine where both critical violations and non-critical stress points are occurring on each of the feeders in their distribution network. For example, Fig. 4.3(a-b) show critical thermal and voltage violations on a single feeder, and Fig. 4.3(c-d) show non-critical three-phase imbalances and power losses, indicating potential cost-saving opportunities.

Based on its historical power flow analysis and its planning and opera-

[3]Demand charges are typically based on a few select hours each year and can account for as much as 40% of a utility's annual energy purchase cost.

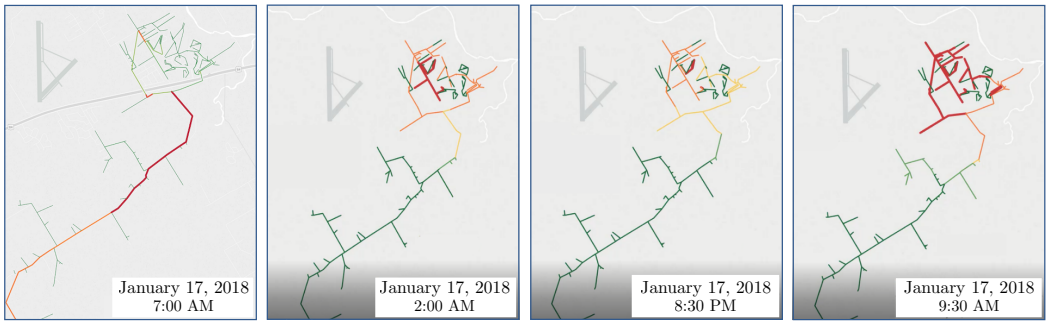

Figure 4.3: Left to right: (a) thermal violations, (b) voltage violations, (c) 3-phase imbalances, and (d) power losses on a real distribution feeder. Red denotes violation of limits, orange/yellow, feeder is nearing its limit and green, the feeder is within normal operating bands.

tional goals, the utility decides where to place a limited number of connections, e.g., 10-20 on each feeder. Note the timestamps in Fig. 4.3, which illustrate that violations and stress points can occur within short time spans (e.g., one day) on a single feeder. Additionally, due to utilities having tens to hundreds of feeders depending on their size, there may be many opportunities for grid support and cost savings.

Concept implementation: The utility would start announcing time-varying rewards at the installed connection ports, serving as incentives for the mobile storage operator as explained below. The rewards are announced the day before and remain fixed the following day when they are being collected.[4]

Next, the mobile storage operator computes an optimal schedule for collecting the best subset of rewards, given available batteries. Note, the

[4]According to our industry contact [117], "day ahead forecasts of grid load are very accurate", so setting effective rewards a day ahead should be viable.

operator can be set up as a profit-maximizing third-party or as an in-house service as part of the utility. Henceforth, we refer to the operator as the 'central planner' or 'planner'.

The third participant in our solution is the group of mobile batteries, which come from: (i) EV owners with variable schedules responding to incentives and (ii) EV fleets owned/managed by ride-sharing companies, cities, autonomous fleets, etc. We envision EV owners would respond to incentives similarly to Uber/Lyft or food delivery drivers. Also, Tesla's Robotaxis and Facedrive are examples of new and expanding EV fleets, which would provide service for monetary compensation and could additionally benefit from the efficient charging and discharge management our solution provides.

To make it worthwhile for all three parties (utility, planner and EVs) to participate, each party needs to be reasonable and share the overall benefit. To this end, the utility's offered rewards need to be less than its expected cost in the absence of the mobile storage solution, say between 25% to 75%. For instance, the utility's expected cost and corresponding nodal cost in the real-world thermal violation from the introduction are \$400,000 and \$225 respectively. At a 40% reward to nodal cost ratio, the utility could announce a reward of \$90 at the designated location(s) and time(s). Additionally, there are voltage violations, demand charges, losses and three-phase balancing that offer cost-saving opportunities to the utility in the \$10 to \$60 range (\$0.13 to \$0.80/kWh).

Benefits and incentives: There are inherent benefits to using EVs over other forms of storage, mainly in that EVs have no upfront or re-occurring expenditures (in the purchase and maintenance of the batteries). So if EVs can be utilized effectively, they would provide notable operating cost savings to both the utility and planner.

For EV owners to participate, they need to have appropriate incentives, which in the simplest case could be free and convenient parking, offering them savings of \$20-\$40 in inner cities over just a few hours. Additionally, owners that do not need parking could still provide their vehicle to the planner for monetary compensation. The EV owners' benefit would thus be: A) free and convenient valet parking, B) a portion of the collected rewards, when offered, and C) the moral benefit of lowering their carbon footprint while also helping lower their and their neighbors' electric bills.

For fleets (e.g., Tesla Robotaxis, city vehicle fleets, rental car fleets, etc.) the incentives would be fixed monetary amounts per discharge and/or profit-sharing. An additional benefit to both EV owners and fleets is that their vehicle will always be returned fully charged, which would be especially attractive to EV owners without in-home charging ability.

The aforementioned benefits should far outweigh the wear and tear on the EV battery, which we estimate at about \$5-\$7 per charging cycle.[5] This estimate is expected to decrease to less than \$2 per cycle with Tesla's launch

[5]This estimate is based on a 5-year replacement of a \$4,000-\$5,000 battery system, with 150 discharging cycles per year.

of the million-mile battery [74]. Furthermore, the planner could allow EV owners to set a limit of one or two discharges per day. This trade-off is similar to Uber/Lyft or food delivery drivers who accept the wear and tear on their cars for more immediate monetary compensation.

In summary, we have made a case that there are opportunities for mitigation of different grid violations along with cost-saving opportunities each day, and when considering each utility's tens or hundreds of feeders, optimal dispatch becomes imperative for a mobile storage solution to be both effective and efficient. Furthermore, we have demonstrated that a mobile storage solution can incentivize EVs to participate at a price point that works for both the utility and the planner. A key advantage of our solution is that it does not lead to a zero-sum outcome and everyone benefits while making the *whole system more efficient* and *increasing social welfare.*

This brings us to the next sections in this chapter, which detail our algorithms (including hardness and performance guarantees) needed to allow for a scalable and effective mobile storage solution.

4.4 Problem Definition and Hardness

In this section we propose our mathematical model for the optimal dispatch of batteries by the mobile storage operator, and study the hardness of the proposed optimization problem.

4.4.1 Problem definition

We consider a set $\mathcal{B} = \{1, 2, ..., m\}$ of *batteries* and a discrete time horizon $\mathcal{T} = [T] = \{1, 2, \ldots, T\}$. Every battery $i \in \mathcal{B}$ is associated with a set $\mathcal{T}_i \subseteq \mathcal{T}$ of *availabilities*, i.e. times where it can be charged and discharged, and a *charging time* $C_i \in \mathbb{N}$, that is, the time the battery needs to recharge before being ready to be discharged again (or returned to the owner). We say that a battery $i \in \mathcal{B}$ is *available* at time t, when $t \in \mathcal{T}_i$. We also consider a set $\mathcal{P} = \{1, 2, \ldots, n\}$ of *ports* that can be used for discharging a battery. At any time $t \in \mathcal{T}$, any available battery $i \in \mathcal{B}$ can be discharged at some port $j \in \mathcal{P}$, and obtain *reward* $p_{j,t} \in \mathbb{R}$. Notice that the reward depends on both the port and time of discharging. Moreover, a reward can be negative, modeling in this way the fact that the benefit to the utility at a specific port/time can be smaller than the cost of recharging. Without loss of generality, we assume recharging is free by redefining each reward to be the difference between the discharging reward and the charging cost. We furthermore assume that the time to move a battery between discharging and charging locations is relatively small and included in the charging time.

Our objective is to find a *feasible* assignment of batteries to ports/times that maximizes the total collected reward. In any feasible assignment, the following conditions must hold:

(a) Every port can be assigned to at most one battery $i \in \mathcal{B}$ at every time $t \in \mathcal{T}$ such that $t \in \mathcal{T}_i$.

(b) Every battery $i \in \mathcal{B}$ can be associated with at most one port $j \in \mathcal{P}$ at any time $t \in \mathcal{T}_i$.

(c) If a battery $i \in \mathcal{B}$ has been discharged at some port at time t, the same battery cannot be discharged again during the time interval $[t+1, t+C_i]$, as it needs C_i time units to get recharged.

Our problem can thus be fully described by the following integer programming (IP) formulation:

$$\max \quad \sum_{i \in \mathcal{B}} \sum_{j \in \mathcal{P}} \sum_{t \in \mathcal{T}} p_{j,t} x_{i,j,t} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{B}} x_{i,j,t} \leq 1, \qquad \forall j \in \mathcal{P}, t \in \mathcal{T} \tag{4.2}$$

$$\sum_{j \in \mathcal{P}} \sum_{t' \in [t, t+C_i]} x_{i,j,t'} \leq 1, \qquad \forall i \in \mathcal{B}, t \in \mathcal{T}_i \tag{4.3}$$

$$x_{i,j,t} \equiv 0, \qquad \forall i \in \mathcal{B}, j \in \mathcal{P}, t \notin \mathcal{T}_i \tag{4.4}$$

$$x_{i,j,t} \in \{0, 1\}, \qquad \forall i \in \mathcal{B}, j \in \mathcal{P}, t \in \mathcal{T}_i. \tag{4.5}$$

In the above IP, $x_{i,j,t}$ denotes the decision variable of discharging battery $i \in \mathcal{B}$ at port $j \in \mathcal{P}$ and at time $t \in \mathcal{T}$. Constraint (4.2) ensures that every port can be used to discharge at most one battery at any time period, while constraint (4.3) is the validity constraint for the recharging restriction, capturing that for every battery i, discharging cannot occur sooner than C_i units of time following the previous discharge.

4.4.2 Hardness result

We now prove that for a non-constant number of batteries and for arbitrary availability intervals, problem (4.1) is *strongly NP-hard*. In terms of algorithmic design, this statement implies that, unless $\mathbf{P} = \mathbf{NP}$, not only is there no polynomial-time algorithm for solving the problem optimally, but also there is no algorithm that can produce an assignment of reward greater than $1 - \epsilon$ of the optimal, in time polynomial in the input size and in $1/\epsilon$.

Theorem 4.4.1. *Problem (4.1) is strongly NP-hard, even for the (easier) special case where all the charging times are identical, and all rewards are 0-1.*

Proof. We use a reduction from the well-known strongly NP-hard 3D-MATCHING problem [118]. In the 3D-MATCHING problem, we are given as input three sets of nodes A, B, C such that $|A| = |B| = |C|$ and a set E of hyperedges of the form $e_i = (\alpha_i, \beta_i, \gamma_i)$, with nodes $\alpha_i \in A, \beta_i \in B, \gamma_i \in C$. The problem is to decide whether there exists a matching $S \subseteq E$ such that every node of $A \cup B \cup C$ is covered by some edge and no two edges of S share a node.

Given an instance $\mathcal{I} = (A, B, C, E)$ of the 3D-MATCHING problem, we begin by considering a time horizon $\mathcal{T}$ such that $\mathcal{T} = [4M + k]$, where $k = |A| = |B| = |C|$ and M is a large number such that $M \geq 2k$. For example, an instance of the 3D-MATCHING problem with $k = 2$ is shown in Fig. 4.4 (right) with its corresponding time horizon on the left (assuming $M = 4$). We create one port j_0 which has reward $p_{j_0,t} = 1$ for times $t \in [1, k] \cup [2M + 1, 2M + k] \cup [4M + 1, 4M + k]$, and $p_{j_0,t} = 0$ for the rest of

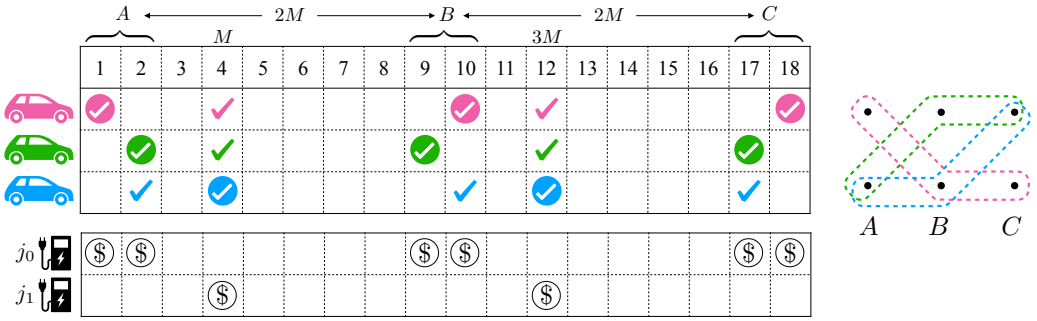

Figure 4.4: Example with $k = 2$, $M = 4$, and $C = 6$, created from the 3D-MATCHING instance on the right. Availabilities of each battery (which encode the hyperedge of the same color) are checked in the upper table, and the circled ones represent the optimal solution which collects all the rewards in the lower table.

the times. Moreover, we create $|E| - k$ additional identical ports, each having reward $p_{j,t} = 1$ for $t \in \{M, 3M\}$, and $p_{j,t} = 0$, elsewhere. For the example of Fig. 4.4, since there are $|E| = 3$ hyperedges, we have only one additional port j_1.

Regarding the batteries, we create a battery i for each hyperedge $e_i = (\alpha_i, \beta_i, \gamma_i)$ and set $\mathcal{T}_i = \{\alpha_i, 2M + \beta_i, 4M + \gamma_i\} \cup \{M, 3M\}$, where we assume that $\alpha_i, \beta_i, \gamma_i \in [k]$ are the indices of the nodes in an arbitrary (fixed) ordering. For example, in Fig. 4.4 the blue hyperedge has $e = (\alpha = 2, \beta = 2, \gamma = 1)$, assuming that the ordering in A, B, C is from top to bottom. Therefore the corresponding battery has availability $\mathcal{T} = \{2, 10, 17\} \cup \{4, 12\}$. Finally, for all batteries, we set the same charging time $C = M + k$.

By the above construction, given a polynomial-time exact algorithm for our problem, one could decide on the feasibility of 3D-MATCHING. More specifically, our algorithm is able to collect all the non-zero rewards if and only

if the answer of the 3D-MATCHING instance is YES. Indeed, the batteries that correspond to the k hyperedges of the matching can collect every reward of the j_0 port, while the remaining $|E| - k$ batteries can collect all the rewards in the $j \neq j_0$ ports. For the other direction, it can be shown that in the case where the answer of the 3D-MATCHING problem is NO, there is no feasible assignment of batteries that can collect all the rewards. □

4.5 Special Tractable Cases

Given that the problem at hand is strongly NP-hard, we turn to designing approximation algorithms. To develop a better understanding and insight into the applicable algorithmic methods, we first analyze several special cases that can be solved optimally in polynomial time.

4.5.1 Zero charging time

When we have access to super-fast chargers, which can charge the battery in a time much smaller than our time unit, we can assume that $C_i = 0$. This implies that any battery can be discharged during any time of its availability, and we show that our problem can be reduced to the MAXIMUM BIPARTITE MATCHING problem [119], which can be solved in polynomial time. To see this, consider a set of vertices L such that $(i, t) \in L$ for any battery $i \in \mathcal{B}$, $t \in \mathcal{T}_i$, and a set of vertices R such that $(j, t) \in R$ for all ports $j \in \mathcal{P}$ and times $t \in T$. We also define the set of edges $E = \{\{(i, t), (j, t')\}, \forall (i, t) \in L, \forall (j, t) \in R, t' = t\}$ and associate every edge that is adjacent to some node $(j, t) \in R$ with *weight*

equal to the reward $p_{j,t}$. Given this construction, the optimal dispatch corresponds to a maximum bipartite matching in the aforementioned graph, which can be solved optimally in polynomial time [119].

4.5.2 Single battery

The problem accepts a polynomial time LP-based algorithm for the case where there is a single battery. In this case, we can assume without loss of generality that there is only a single discharging port, otherwise we consider the port of highest reward at every time. We should note that, although the case of a single battery is contained in the case of constant number of batteries (studied next), the following result provides useful intuition on the geometry and polyhedral aspects of the problem. Let $x_t \in \{0, 1\}$ be the decision variable denoting the discharging of the single battery at time t. Consider the following LP relaxation of our problem:

$$\max \quad \sum_{t \in \mathcal{T}} p_t x_t \tag{4.6}$$

$$\text{s.t.} \quad \sum_{t' \in [t, t+C_1]} x_{t'} \leq 1, \qquad \forall t \in \mathcal{T}_1 \tag{4.7}$$

$$x_t \equiv 0, \qquad \forall t \notin \mathcal{T}_1 \tag{4.8}$$

$$x_t \geq 0, \qquad \forall t \in \mathcal{T}. \tag{4.9}$$

We can show the following important property about the feasibility region of the above LP.

Theorem 4.5.1. *All the vertices of the polytope (4.7-4.9) are integral.*

Proof. Constraint (4.7) can be written in matrix form as $Ax \leq b$, in which A is a $\{0,1\}$-matrix, and b is an all-ones vector. Note that every row of A has $1+C_1$ consecutive ones, and the remaining entries are all zeros. Such a matrix is called an interval matrix, a special class of totally unimodular matrices (see, e.g. [120] Chapter 19). An alternative characterization of a totally unimodular matrix A is that for all integral vectors a, b, c, d, the polyhedron $\{x | c \leq x \leq d, a \leq Ax \leq b\}$ has only integral vertices. Since our constraints (4.7-4.9) are of such form, we conclude that all the extreme points of the LP (4.6-4.9) are integral. □

Since the LP relaxation provides an upper bound, the above integrality result implies that the solution of the LP is the optimal solution to the original problem.

4.5.3 Constant number of batteries

The third class of instances that can be solved in polynomial time corresponds to the case where the number of batteries (m) is small enough and can be considered constant. For this case, we can solve the problem using dynamic programming. More specifically, we construct a matrix $OPT \in \mathbb{N}^{m+1}$ with entries $OPT(t, r_1, r_2, \ldots, r_m)$, where every variable r_i denotes the remaining time for a battery $i \in \mathcal{B}$ until it is able to be discharged again. Notice that the total size of this matrix is $\mathcal{O}(TC_{\max}^m)$, where $C_{\max} = \max_{i \in \mathcal{B}} C_i =$

$\Theta(T)$. In general, for the computation of every element $OPT(t, r_1, r_2, \ldots, r_m)$ of the matrix, we want to find which combination of available batteries we can discharge such that the reward collected plus the future optimal reward given this discharging is maximized. Notice that trying all possible combinations of batteries to discharge is polynomial, given the assumption of constant m. By the above analysis, it suffices to recursively compute all the entries of the $OPT(t, r_1, r_2, \ldots, r_m)$ matrix, starting from the end of the time where the solution is trivial. After the computation of the whole matrix, the value $OPT(t = 1, r_1 = 0, r_2 = 0, \ldots, r_m = 0)$ would be the answer to our problem. Notice that we can easily extend the above algorithm to keep track of our best choices at each time and return, in addition to the optimal reward, the optimal dispatch. In the following, we give an example of a recursive computation for the case of a single port:

$$OPT(t, r_1, r_2, \ldots, r_m) = \max_{i \in \mathcal{B} \cup \{\emptyset\}} \Big\{ OPT(t+1, C_i, r_{\bar{i}} - 1) + p_{j,t} \cdot \mathbf{1}[t \in \mathcal{T}_i] \Big\},$$

where $OPT(t+1, C_i, r_{\bar{i}}-1)$ is the short form for $OPT(t+1, r_1, r_2, \ldots, r_m)$ where we replace r_i with C_i and all other indices (denoted by $\bar{i}$) are decremented by one (if they are positive).

4.5.4 Homogeneous batteries

A similar dynamic programming approach works for the case where all batteries have the same availability period and the charging time, say C, is constant and identical. In that case, we only care about the number of available batteries and not their identity. We construct a matrix $OPT \in \mathbb{N}^{C+2}$,

with elements $OPT(t, r_0, r_1, r_2, \ldots, r_C)$, where $t \in \mathcal{T}$ is the time and r_ℓ is the number of batteries that will be available in ℓ time periods. Since ℓ is upper bounded by the charging time C, the size of the matrix is $\mathcal{O}(Tm^{C+1})$, and the elements can again be computed in a recursive manner as follows:

$$OPT(t, r_0, r_1, \ldots, r_C) = \max_{0 \le k \le r_0} \Big\{ OPT(t+1, r_0 + r_1 - k, r_2, \ldots, r_C, k) + \max_{|S|=k} \sum_{j \in S} p_{j,t} \cdot \mathbf{1}[t \in \mathcal{T}_1] \Big\},$$

where k is the number of batteries that we dispatch at time t, upper bounded by the number of available batteries r_0. For the optimal k, the set S corresponds to the top k ports with the highest rewards, which will be assigned to those k batteries. In the next time step $t+1$, these batteries will appear as the last argument, since they now need C time steps to get charged. In addition, all other arguments will be shifted one to the left, as the batteries which needed ℓ time periods (for charging) at time t will need another $\ell - 1$ time periods at time $t+1$.

4.6 Approximation Algorithms

Building on the above, in this section we present three approximation algorithms for our optimization problem (4.1).

4.6.1 A $\frac{1}{3}$-approximation greedy algorithm

We first present a greedy algorithm. Even though this algorithm has a weaker approximation guarantee than the algorithm in the following sec-

Algorithm 5 GREEDY

1: Initialize: $\mathcal{F} = \{(i, j, t) | i \in \mathcal{B}, j \in \mathcal{P}, t \in \mathcal{T}_i\}$.
2: **while** $\mathcal{F} \neq \emptyset$ **do**
3: Choose the feasible assignment of maximum reward: $(i, j, t) \leftarrow \text{argmax}\{p_{j,t} | (i, j, t) \in \mathcal{F}\}$.
4: Assign battery i to the port/time (j, t) and collect the reward $p_{j,t}$.
5: Remove from the feasible set $\mathcal{F}$ all the assignments that are no longer feasible.
6: **end while**

tion, it has the advantage that it applies to a more general setting where the rewards can depend not only on the time and location, but also on the battery. That is, for any choice of battery $i \in \mathcal{B}$, port $j \in \mathcal{P}$ and time $t \in \mathcal{T}_i$ the collected reward can be battery specific, denoted by $p_{i,j,t}$. We define $\mathcal{F} = \{(i, j, t), i \in \mathcal{B}, j \in \mathcal{P}, t \in \mathcal{T}_i\}$ to be the set of all feasible assignments, that is, all the possible triplets of batteries to port/time pairs that can be realized individually, assuming an empty schedule. Consider the following algorithm: at each iteration, we greedily pick the highest available reward, assign it to an available battery i (or the battery that gives that reward in the case of battery-dependent rewards), and remove all the triplets that are no longer feasible. This includes any other battery that wants to discharge at the same port and time, as well as the triplets that wish to discharge the same battery within a period of C_i time intervals. The formal algorithm is tabulated under Algorithm 5. We can prove the following performance guarantee for it:

Theorem 4.6.1. *The* GREEDY *algorithm produces in polynomial time a solution of total reward* $P \geq \frac{1}{3}OPT$*, where* OPT *is the reward of an optimal*

dispatch.

Proof. The polynomial running time of the algorithm can be readily justified. The algorithm sorts $\mathcal{O}(mnT)$ initial assignments and then constantly updates the ordered list in linear time, by removing any infeasible assignments. The total number of iterations is at most $\mathcal{O}(\min(nT, mT))$. On the approximation ratio of the algorithm, it suffices to make the following crucial observation: Let A and O be the set of assignments in the GREEDY and the optimal solution, respectively. For any assignment $(i, j, t) \in A$, we distinguish between two cases: whether (a) it is in the optimal solution $(i, j, t) \in O$ or (b) it is not, i.e., $(i, j, t) \notin O$. Notice that in the second case, the assignment (i, j, t) excludes at most three assignments that are made in the optimal solution: (b1) $\exists i' \neq i$ s.t. $(i', j, t) \in O$, that is, another battery is assigned to the same port/time pair in the optimal solution, (b2) there exist $t' \neq t$ and j' such that $(i, j', t') \in O$ and $t' \in [t - C_i, t + C_i]$, i,e, the optimal solution contains an assignment of the same battery i that is incompatible with (i, j, t) given the charging time restriction. It is not hard to verify that for any battery i, there can be at most two assignments in any interval $[t-C_i, t+C_i]$. Therefore, given that GREEDY always chooses the assignment of highest reward, the reward of every $(i, j, t) \in A$, $p_{j,t}$, is greater than $\frac{1}{3}$ times the sum of rewards of the

assignments of the optimal solution it excludes. Therefore, we have:

$$\begin{aligned} P &= \sum_{(i,j,t)\in A} p_{j,t} \\ &= \sum_{(i,j,t)\in O\cap A} p_{j,t} + \sum_{(i,j,t)\in A\setminus O} p_{j,t} \\ &\geq \sum_{(i,j,t)\in O\cap A} p_{j,t} + \frac{1}{3}\sum_{(i,j,t)\in O\setminus A} p_{j,t} \\ &\geq \frac{1}{3}\Big[\sum_{(i,j,t)\in O\cap A} p_{j,t} + \sum_{(i,j,t)\in O\setminus A} p_{j,t}\Big] \\ &= \frac{1}{3}\sum_{(i,j,t)\in O} p_{j,t} = \frac{1}{3}OPT. \end{aligned}$$

□

4.6.2 A $\frac{1}{2}$-approximation greedy algorithm

Intuitively, a major issue with the algorithm GREEDY, is that the choice of every step is oblivious to the charging time restrictions for every battery. For example, consider an instance of a single battery and three possible assignments $p_{i,j_1,t=1}, p_{i,j_1,t=C_i}, p_{i,j_1,t=2C_i}$, such that $p_{i,j_1,t=1} = p_{i,j_1,t=2C_i} = 1$ and $p_{i,j_1,t=C_i} = 1+\epsilon$, for a really small $\epsilon > 0$. In that case, the GREEDY would choose the assignment of reward $1+\epsilon$, making the other two incompatible, while the optimal choice is of reward 2, and consists of choosing the first and the third feasible assignment. In order to improve this situation, we propose the following alteration of the above greedy algorithm, namely, GREEDY+. The new algorithm starts from an arbitrary battery and finds the optimal allocation for this battery individually, which, as we have described, can be done

Algorithm 6 GREEDY+

1: Initialize: $\mathcal{F} = \{(i, j, t) | i \in \mathcal{B}, j \in \mathcal{P}, t \in \mathcal{T}_i\}$.
2: **while** $\mathcal{F} \neq \emptyset$ **do**
3: Choose a new arbitrary battery $i \in \mathcal{B}$ and compute the most profitable combination of assignments, given the set of feasible assignments $\mathcal{F}$.
4: Assign the computed pairs of port/time to battery i and collect the corresponding rewards $p_{j,t}$.
5: Remove from the feasible set $\mathcal{F}$ all the assignments that are no longer feasible.
6: **end while**

in polynomial time. After this, the corresponding rewards are collected and the incompatible assignments are removed. The procedure is repeated over all batteries. The algorithm is formally stated in Algorithm 6.

We are now able to prove a stronger approximation guarantee for GREEDY+:

Theorem 4.6.2. *The* GREEDY+ *algorithm produces an assignment of total reward* $P \geq \frac{1}{2}OPT$*, where* OPT *is the reward of an optimal assignment.*

Proof. We denote by $S_{i,i'}$ the set of pairs $(j, t), j \in J, t \in \mathcal{T}$ such that the optimal schedule assigns them to $i \in \mathcal{B}$ while the schedule produced by GREEDY+ assigns them to $i' \in \mathcal{B}$. Let also OPT_i and P_i be the total reward collected by battery $i \in \mathcal{B}$ in the optimal and in our schedule, respectively. Clearly, $OPT = \sum_{i \in \mathcal{B}} OPT_i$ and $P = \sum_{i \in \mathcal{B}} P_i$, where P is the total reward collected by our algorithm. At the first iteration, our algorithm chooses a battery, say $i_1 \in \mathcal{B}$, and computes in polynomial time the optimal assignments of this battery alone, in which case $P_{i_1} \geq OPT_{i_1}$. For the rest of the batteries, our algorithm collects reward: $P_i \geq OPT_i - \sum_{i' \in [i]} \sum_{(j,t) \in S_{i,i'}} p_{j,t}$. Intuitively, at

every iteration the chosen battery collects reward at least equal to the reward it collects in the optimal schedule minus the rewards that are already collected in previous rounds by other batteries. Adding all these inequalities, we get:

$$P = \sum_{i \in \mathcal{B}} P_i \geq \sum_{i \in \mathcal{B}} \left(OPT_i - \sum_{i' \in [i]} \sum_{(j,t) \in S_{i,i'}} p_{j,t} \right)$$
$$\geq \sum_{i \in \mathcal{B}} OPT_i - \sum_{i \in \mathcal{B}} \sum_{i' \in [i]} \sum_{(j,t) \in S_{i,i'}} p_{j,t}.$$

Notice that the second term in the right hand side satisfies

$$\sum_{i \in \mathcal{B}} \sum_{i' \in [i]} \sum_{(j,t) \in S_{i,i'}} p_{j,t} \leq P,$$

that is the total amount of reward "stolen" by other batteries cannot be more than the total collected reward. After plugging this in the above inequality, the theorem follows. □

4.6.3 A $(1 - \frac{1}{e})$-approximation randomized algorithm

In this section, we present an LP-based randomized algorithm with a performance guarantee of $1 - \frac{1}{e} \approx 0.63$, which is better than the previous greedy algorithms. We consider the linear programming relaxation of IP (4.1), which we create by allowing the decision variables to take any non-negative value, i.e. $x_{i,j,t} \geq 0$.

This LP formulation gives an upper bound to the optimal solution of our problem. Since the LP solution in not necessarily integral (or even half-integral), we adapt a randomized rounding algorithm, proposed in [121]

for an interval scheduling problem with application to bandwidth trading. Although our problem is not the same, nor a special case of the bandwidth trading problem [121], they both turn out to be special cases of a more general problem, which can be solved by the same randomized rounding algorithm. The formal description of the algorithm adapted to our problem is given under Algorithm 7.

Since the LP solution can be fractional, a single battery could be assigned fractionally to conflicting discharging ports. But constraint (4.3) guarantees that at each time, the total assignment is at most one, allowing us to fit the assignments in a strip of height one as shown in Fig. 4.5. The rounding phase consists of sampling a random horizontal line in this figure, and picking the assignments that it intersects.

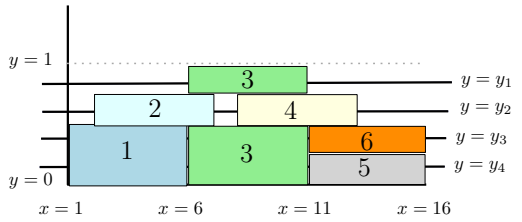

Figure 4.5: Example of Randomized Rounding for a fixed battery i of $C_i = 4$. We have the following non-zero assignments: $x_{i,1,1} = 0.50$, $x_{i,2,2} = 0.25$, $x_{i,3,6} = 0.75$, $x_{i,4,8} = 0.25$, $x_{i,5,11} = 0.25$ and $x_{i,6,11} = 0.25$. Notice that the rectangle of port 3 has been fragmented into two slices of height 0.25 and 0.50. For the random horizontal lines $y = y_1$ to $y = y_4$, the corresponding feasible assignments are $\{3\}$, $\{2, 4\}$, $\{1, 3, 6\}$ and $\{1, 3, 5\}$, respectively.

The following theorem provides a performance guarantee for the randomized rounding algorithm. We refer the reader to [121] for a complete proof.

Algorithm 7 RANDOMIZED ROUNDING

Solve the LP relaxation and let $\{x_{i,j,t}\}$ be the (fractional) solution.
for every battery $i \in \mathcal{B}$ **do**
    Let $A_i = \{(j,t) | j \in \mathcal{P}, t \in \mathcal{T}_i, x_{i,j,t} > 0\}$
    Consider an empty 2-dimensional area: $[1, T+1] \times [0,1] \subseteq \mathbb{R}^2$ defined by the x-axis and the y-axis.
    for $(j,t) \in A_i$ in non-decreasing order of $t \in \mathcal{T}$ **do**
        Create a rectangular area of y-coordinates within $[0, x_{i,j,t}]$ and of x-coordinates within $[t, t + C_i + 1)$, if the corresponding area is empty.
        If this area is not empty, split the rectangle into smaller stripes of fixed width $[t, t + C_i + 1)$, and fit these slices anywhere in the y-axis. (Notice that the rectangles can be fragmented only in the y-axis, but no fragmentation is allowed in the x-axis.)
    end for
    Sample a horizontal line from $y = 0$ to $y = 1$ uniformly at random.
    Assign battery $i \in \mathcal{B}$ to all the ports and times which are crossed by the line.
end for
If more than one battery has been assigned in the same $(j,t), j \in \mathcal{P}, t \in \mathcal{T}$, arbitrarily keep one.

Theorem 4.6.3 ([121]). *The* RANDOMIZED ROUNDING *algorithm creates a feasible assignment of expected reward* $\mathbb{E}[P] \geq (1 - \frac{1}{e})OPT$*, where* OPT *is the reward of an optimal assignment.*

4.7 Simulation Results

In this section, we evaluate the empirical efficiency of our proposed algorithms. In the absence of actual data on our problem and in order to prove the robustness of our algorithms, we choose test cases of variable charging times, variable availability intervals and rewards that are generated by random

initial starting values and then randomly change at each time step within a defined range.

4.7.1 Simulation setting

We compare the performance of our algorithms with the optimal reward OPT (which we obtain by solving the IP) or with an upper bound on OPT (which we obtain by solving the LP relaxation). We also consider the *boosted* version of the randomized algorithm, called BOOSTED RANDOMIZED ROUNDING, in which we repeat the randomized algorithm 10 times and return, among the solutions we found, the one with the highest reward, amplifying in this way the probability of choosing a profitable dispatch.

In terms of test cases, we consider a time horizon of $T = 24$ hours and various combinations of the number of batteries and ports. The charging time of each battery is a number between 1 and 6 hours (to capture different capacities, charging rates, etc.) chosen uniformly at random (u.a.r.). On the availability of each battery, we distinguish between two types: those with a single continuous interval of length 1 to 24 chosen u.a.r., and those with three (smaller) intervals of length 1 to 8 chosen u.a.r.. We also define the starting time of each interval to be a number between 1 and 24 chosen u.a.r.. On the choice of rewards, we initially choose for each port $j \in \mathcal{P}$ a value $p_{j,1}$ between 0 and 100 u.a.r. to denote the average reward, differentiating in this way the ports in terms of their location. Then, we model a smooth change in the reward on a specific port as follows: for $t > 1$ we draw $p_{j,t}$ from $[\ell_{j,t}, u_{j,t}]$ u.a.r.

with $\ell_{j,t} = \max\{0.7 \cdot p_{j,t-1}, p_{j,1} - 25, 0\}$ and $u_{j,t} = \min\{1.3 \cdot p_{j,t-1}, p_{j,1} + 25, 100\}$. That is, the reward of any (hour, port) pair is always within 70% to 130% of the reward of the previous hour, and always within a band of ± 25 from the initial reward.

Given our observation that the number of batteries is without loss of generality greater that the number of ports, we simulate for different values of the following two parameters: (a) The number n of discharging ports and (b) the *ratio* of the number of batteries m, over the number of ports n, denoted by $R = \frac{m}{n}$. The values we consider are $n \in \{1, 5, 10, 20, 50, 100, 200\}$ and $R \in \{1, 2, 4, 8\}$. Finally, for every choice of n and R, we perform 10 independent experiments and return, for each algorithm, the average of the empirical ratios found.

We solve the integer and linear programs using Gurobi Optimizer 8.0, while we use Python 2.7 for scripting.

4.7.2 Presentation and analysis of the results

The empirical approximation ratios resulting from our simulations are presented in Table 4.1 and Fig. 4.6. We denote by star (*) the cases where the approximation ratio is computed with respect to the optimal reward.

In Fig. 4.6 we have plotted the performance ratio of the proposed algorithms separately for each value of R, the ratio between the number of batteries and ports as a function of n. There is no clear relation between the number of ports (n) on the x-axis and the performance ratio on the y-axis (remem-

Table 4.1: Empirical approximation ratios.

R=1	1*	5*	10*	20*	50*	100*	200*
G	0.888	0.898	0.876	0.872	0.873	0.871	0.880
G+	1.000	0.988	0.991	0.993	0.997	0.998	0.999
RR	1.000	0.993	0.963	0.937	0.913	0.908	0.826
BRR	1.000	0.993	0.966	0.960	0.959	0.939	0.919
R=2	1*	5*	10*	20*	50*	100*	200*
G	0.931	0.878	0.889	0.889	0.883	0.889	0.883
G+	0.990	0.981	0.987	0.994	0.995	0.997	0.998
RR	1.000	0.971	0.968	0.894	0.875	0.866	0.883
BRR	1.000	0.982	0.977	0.942	0.922	0.935	0.928
R=4	1*	5*	10*	20*	50*	100*	200*
G	0.872	0.900	0.909	0.905	0.909	0.909	0.911
G+	0.955	0.977	0.983	0.990	0.993	0.996	0.997
RR	0.964	0.983	0.898	0.876	0.847	0.861	0.870
BRR	0.967	0.983	0.948	0.938	0.926	0.929	0.933
R=8	1*	5*	10*	20*	50*	100	200
G	0.923	0.951	0.961	0.961	0.964	0.965	0.966
G+	0.973	0.986	0.989	0.992	0.995	0.997	0.998
RR	0.988	0.967	0.962	0.979	0.981	0.978	0.981
BRR	0.988	0.982	0.977	0.986	0.989	0.992	0.993

ber that our worst-case performance ratios are constant). Hence, these plots should be merely seen as a comparison between the proposed algorithms in different independent scenarios. Based on our simulation results, we can make the following observations:

- The actual performance of our algorithms in every case is significantly better than the theoretically proven worst-case guarantees, i.e. $82\% - 100\%$ of the optimal. Moreover, the overall performance of all algorithms increases with the ratio R. Therefore, for a fixed number of ports, the

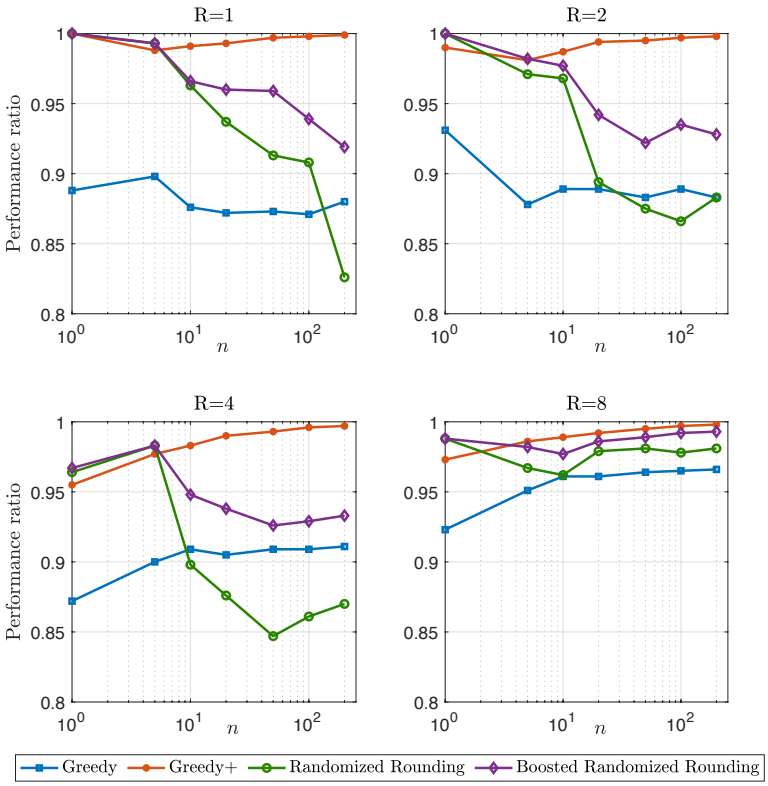

Figure 4.6: Performance ratio of the proposed algorithms for different values of $R = m/n$ (number of batteries over the number of ports).

performance of our algorithms gets closer to the optimal solution as we employ more batteries.

- The GREEDY algorithm (which has the worst approximation ratio of $\frac{1}{3}$), has the worst empirical performance among our algorithms, specifically $87\% - 96\%$ of the optimal.

- For the case of $R = 1$ and $n = 1$, we observe that the RANDOMIZED ROUNDING always produces the optimal result, a fact that is justified by

the integrality of the LP relaxation in this case (see Theorem 4.5.1).

- The GREEDY+ algorithm achieves the best performance, producing empirical ratios of at least 95% of the optimal. After that, BOOSTED RANDOMIZED ROUNDING has the next highest empirical performance of at least 92%, which is significantly better than the simple RANDOMIZED ROUNDING.

4.8 Conclusion

The distribution grid is ripe for cost-effective storage solutions. We have proposed one such solution that utilizes EVs to provide grid support. As shown in the real-world example in the introduction, our mobile storage solution can be more flexible and cost-effective than traditional line/equipment upgrades or stationary storage, while still accomplishing a variety of planning and operation objectives. Our solution can add a new layer to a utility's toolkit, while also helping the transition of distribution planning and operations into a future of notably higher EV and renewable grid penetration.

Our contributions here have included a description of our mobile storage concept and the supporting theory to bring it to fruition, including a new computational hardness result and efficient algorithms with provable guarantees that also perform very well empirically. Future work could include extending the current theoretical model to online EV arrivals as well as mechanism design for aligning utility rewards with EV participation incentives. Another interesting direction would be to investigate combining EV ride-sharing with EV grid support.

Chapter 5

Energy Procurement with Abandonment

In this chapter, we study a dynamic model of procurement auctions which is motivated by the procurement of energy. In the energy sector, if the system operator does not assign sufficient generation to a specific energy generator, the generator may go out of business due to their overhead costs and financial constraints. Motivated by this, we study a dynamic model of procurement auction in which the agents (sellers) will abandon the auction if their utility does not satisfy their private target, in any given round. We call this "abandonment" and analyze its consequences on the overall cost to the mechanism designer (buyer), as it reduces competition in future rounds of the auction and drives up the price. We show that in order to maintain competition and minimize the overall cost, the system operator may choose to assign the generation to less efficient sources in order to make sure that they survive and make it to the future rounds. We focus on threshold mechanisms as a simple way to achieve ex-post incentive compatibility, akin to reserves in revenue-maximizing forward auctions. We then consider the optimization problem of finding the optimal thresholds. We show that even though our objective function does not have the optimal substructure property in general, if the underlying distributions satisfy some regularity properties, the global optimal

solution lies within a region where the optimal thresholds are monotone and can be calculated with a greedy approach, or even more simply in a parallel fashion.[1]

5.1 Introduction

The wide applicability of auctions in real life, from the simple traditional sealed-bid and ascending/descending price auctions, to the modern sponsored search and eBay auctions, to government-run auctions for spectrum and carbon emissions, has inspired the development of a rich theory of auctions and mechanism design. The more prevalent auction design focuses on the so called 'regular' auctions, where the bidders are buyers wishing to buy an item from the mechanism designer (seller), who tries to maximize her revenue (see, e.g. [123]). Less prevalent are 'reverse' or 'procurement' auctions where the bidders are sellers and the mechanism designer is a buyer wanting to minimize cost.

A principal example of procurement auctions is public procurement—the process by which governments purchase goods, services and construction—which comprises a significant fraction, 10-20%, of a country's GDP [124]. Some of the more complex procurement auctions include the above mentioned spectrum and carbon emissions auctions, as well as the procurement of energy, a

[1]This chapter is based on the manuscript submitted to the ACM Conference on Economics and Computation [122]. The author of this dissertation made the primary technical contribution to this work.

key motivation of this chapter.

A notable feature of the above examples that makes the corresponding auction processes especially complex in reality is their repeated nature with interdependencies among the different rounds. Thus, while the large majority of literature on auction and mechanism design focuses on *static* mechanisms that optimize the designer goals with a single round in mind, there has been a recent rise in the study of *dynamic mechanism design* which attempts to model and analyze mechanisms across time [125]. In the context of procurement, different strands of literature investigate different types of interdependencies, such as caused by a capacity constraint [126, 127], a switching cost from one service provider to another [128, 129], a backlog cost in dynamic inventory control models [130, 131], learning through experience [132, 133] and piecewise procurement where the subprojects of a large project have to be procured in a predetermined order [134, 135].

Yet very simple and basic models for dynamic procurement remain unexplored that provide fertile ground for theory exploration and progress. We propose one such model which takes the most basic reverse auction of multiple sellers needing to provide a unit of divisible good or service over repeated rounds, with the condition that a seller must make at a minimum her overhead cost in order to remain present in future rounds of the auction. This provides a coupling or interdependency of the different rounds of the auction that precludes existing mechanisms from applying and calls for new tools in mechanism design.

Our motivation for this model comes from the process for energy procurement called "economic dispatch": Electricity generation is currently managed by Independent System Operators (ISO) in a myopic way (day by day). Each generator submits a supply curve, namely one or more bids of how much it is able to generate at what unit cost, for the following day. The ISO then allocates generation, based on demand (and subject to any system constraints), so as to minimize the total generation cost. Economic dispatch is thus effectively a generalized version of the standard procurement auction.

In a lot of US markets, wind is typically the least expensive form of generation, thus it is favored by the current selection mechanism over conventional generation (nuclear, coal and gas). Coal, as the least competitive conventional generation, is gradually being driven out of business due to underuse. Wind though has higher variability and uncertainty, and requires increased use of expensive back-up generation, while conventionals are reliable and do not need to be backed up. Ultimately, this is pushing the system to the two extremes of cheap, variable renewables and expensive, back-up generation. As a result, this short-term cost-minimization approach yields a higher long-term cost and compromises system reliability [136].

In reality, a less competitive generator whose economic viability is threatened might be "saved" by the ISO if it is considered critical to system reliability, by entering a side contract with the ISO that guarantees it sufficient allocation and payment to help it remain viable. Such contracts are currently done behind closed doors in an ad hoc way, including the ISO's

decision which generators it considers critical.

To improve system efficiency and transparency, our model here makes a first step toward providing a framework for systematic allocation and payments that minimize cost over multiple periods. Specifically, two issues stand out from the brief background on energy procurement above. One issue is the need to capture the agents' overhead costs necessary to stay in business as a model feature to make transparent the process of identifying and saving a needed agent. We call the phenomenon of permanently leaving the auction due to not having met the overhead cost in a given round as "abandonment". The second, related issue, is the tension of cost vs competition, or short- vs long-term outlook, namely that being optimal in the current round might be suboptimal from a long-term perspective. That is because cost minimization in a given round might result in fewer agent allocations and thus reduced competition in future rounds, which would lead to a higher cost in the future. We discuss these two issues in more detail in the context of our model and results below.

Modeling choices and assumptions. Our goal is to frame the above real-life situation as a simplified auction theory model that abstracts away many engineering components, which are important but not central to the core mechanism design challenges. What are the minimal features our model can be stripped down to, that make it as simple as possible yet expressive of the two above-mentioned issues of (i) abandonment and (ii) tradeoff of cost and

competition?

We focus on a two-round model with n symmetric agents (sellers), each of whom can meet the entire demand of 1 unit of divisible good/service per round, and each of whom submits a bid for her overhead cost, namely the amount she needs to make this round to "be saved" and remain in the next round of the auction. The overhead costs are private values, independent and identically distributed according to some known distribution F, across agents and across rounds.

To keep the model tractable, we assume that a per unit production cost that sellers incur for providing the good/service is known and constant (which turns out mathematically equivalent to it being zero). For example, in the energy application above, the cost of producing energy can easily be estimated by the technology; however, the overhead costs of generators (such as financing, labor costs, maintenance, etc.) are private information.

In a given round, the auctioneer or mechanism designer collects the bids and decides on the allocations and payments which in turn determine which agents are going to be saved for the following round. We will argue later that in the last day, the mechanism designer is going to allocate the entire demand to a single agent (since there is no need to maintain the competition anymore). Further, to more succinctly capture the challenges that abandonment and competition issues present, we assume that even in that final round, the mechanism should satisfy the overhead cost constraint of that single agent: this is also equivalent to removing this assumption and having

one extra round, namely a 3-round auction setting.

Abandonment. In both forward and reverse auctions, when the auction is repeated over several iterations, it has been noted that the agents may leave the platform. The typical assumptions used in the literature are of dynamic arrival and departure that are exogenous and are not related to the outcome of the auction [137, 138]. In a regular auction the agent may prefer to change her auction platform if she is not receiving enough utility. Similarly, in a reverse or procurement auction, for example in the energy sector, if the generators do not meet their overhead costs they are forced to close down.

Two natural modeling choices for the utility function of an agent stand out to capture the abandonment: the utility for being allocated zero or, more generally, for being paid less than one's overhead cost could be modeled as zero or as negative infinity (or, equivalently, a large negative constant). The first choice may appear more natural on the surface but it fails to align the incentives with the phenomenon of abandonment—specifically, it fails to represent the negative repercussions of a bankruptcy in reality, which is what we are trying to model with agents abandoning the auction. Indeed, if an agent ever goes out of business, the agent should not be incentivized to stay in the auction. Furthermore, zero utility for zero allocation is inaccurate in the energy context where power plants continue having overhead expenses (such as employee salaries and power plant maintenance) even if they are not allocated and not producing in a given time period, so effectively a zero or even insuffi-

cient positive allocation implies losses which are what ultimately drives plants to retire. We thus opt for the negative infinity model, which also emphasizes the "finality" of an agent's participation in the auction if she is not allocated or has not met her overhead cost in a given round.

We remark that with this modeling choice, the utility function will not satisfy individual rationality, in that participating in the auction may have lower expected utility than not participating. Again, this is consistent with the energy and likely a number of other applications where starting a business such as building and operating a power plant entails risk and is not guaranteed to break even. We note the relation of our utility function choice to regular auctions where a buyer has a budget and receives a utility of negative infinity for exceeding it (e.g., [139, 140]). Indeed, we can view the overhead cost that needs to be met each period as a reverse budget where, once the budget is exceeded, or in our case the reverse budget is not met, the agent is forced to abandon the auction.

Competition vs cost. In a one-shot environment it is well understood that competition lowers cost and a monopoly increases it. Over multiple rounds, however, the connection of cost and competition is not as straightforward. Already in our two-round setting, we can see that in order to meet the demand while satisfying agent overhead costs, it is cheapest for the mechanism designer to pick the single lowest bid agent and allocate the entire demand to her—cheapest in the first round, that is. This can be implemented by running a

Vickrey-Clarke-Groves (VCG) mechanism where we ask each agent for their overhead cost. However, due to abandonment, this would result in all but one agent abandoning the auction after the first round, and would lead to a monopoly situation in the second round whereby the designer needs to pay the maximum amount to the single remaining agent—paying the upper bound of the agent's bid distribution. Allocating to multiple agents in the first round would be a lot more costly in that round: is the higher cost worth the savings that would result from increased competition in round 2? Indeed this question highlights one of the central design challenges for our mechanism discussed in Section 5.4, and a key technical challenge in the resulting multi-variate optimization problem discussed in Section 5.5.

Bid-sensitive vs bid-oblivious mechanisms. Following the above discussion, there is a trade-off between the cost we incur on day 1, and the competition that exists on day 2. We thus expect that there is an optimal point in the trade-off between cost and competition. In other words, there should be an optimal number of agents that balances the cost required to save those agents today, and the expected cost we incur tomorrow given the competition among this surviving number of agents. Let k denote this optimal number of agents. Now we can generalize the well known truthful second-price auction as follows: after the agents submit their bids, the mechanism assigns the demand to the cheapest k agents and all those k agents receive a payment equal to the $(k+1)$st bid. Note that the higher k is, the higher the cost would be in the

current round, but there will be more competition (hence less cost) in future round(s), and vice versa. Hence, the optimal k can be calculated based on the distribution F and the number of rounds left.

In fact, we can do even better than the mechanism described above, which we call a "bid-oblivious" mechanism. In a bid-oblivious mechanism, we commit to save k agents, independently of their bids. Instead, we could let the bids determine the number of agents to be saved, via using appropriate thresholds. This is similar to revenue maximization in regular auctions, in which by setting a reserve price we ensure that the item is not sold at a cheap price. If all the bids happen to be less than the reserve price, the item would not be sold. Similarly here, we can save k agents only if the lowest k bids are below a certain threshold. In this way, we can dynamically decide on the number of agents to be saved, and achieve a lower cost. We call this a "bid-sensitive" mechanism, as the bids affect the number of agents that get allocated the service. We provide an example on how bid-sensitive mechanisms can achieve a lower cost compared to bid-oblivious mechanisms in Appendix C.1.

Preview of our results and challenges. Our goal is to design the optimal bid-sensitive mechanisms, i.e., to find the optimal thresholds for allocating the service to various number of agents at every round. Specifically, for our two-round auction, it suffices to set the corresponding thresholds for round one, as the final round has a trivial optimal mechanism. We denote by t_i the threshold for saving i agents in round one. Our main result is to show that the global

optimization for the thresholds $t_2, t_3, ...$ can be done in a greedy fashion, even though the objective function does not have the optimal substructure property that usually leads to optimality of greedy approach (see Theorem 5.5.5).

To illustrate this, consider and example with three agents in which the overhead costs are drawn from a uniform $[0, 1]$ distribution. Let $C(t_2, t_3)$ be the expected cost of both rounds, given that we set thresholds t_2, t_3 for round one (meaning that we save all three agents if all 3 bids are below t_3; otherwise, we save two agents if 2 bids are below t_2, and if not we save only one agent. Note that $t_1 = 1$, since we always have to save at least one agent). We show in Example 2 in Section 5.4 that the optimal thresholds in this setting are $(t_2^*, t_3^*) = (\frac{1}{6}, \frac{1}{12})$. Yet this simple example reveals a few interesting observations:

1. If the underlying distribution satisfies certain properties, the optimal thresholds are monotone, i.e., $t_2^* \geq t_3^* \geq t_4^* \geq \cdots$ (see Lemma 5.5.1 and Theorem 5.5.5 in Section 5.5). Even though this may seem intuitive, we show in Example 5 in Appendix C.4 that this is not always true for general distributions.

2. We can find the optimal threshold mechanism with a greedy algorithm, meaning that we can start by optimizing over t_2 (while setting $t_3, t_4, ...$ to any arbitrary values). This gives the optimal value for t_2. Then, using the optimal value of t_2, we can optimize over t_3. However, if we do this process in any other order, it does not result in the optimal threshold

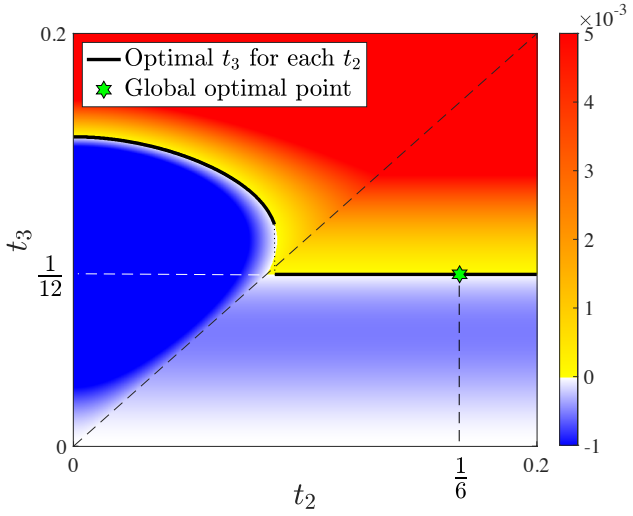

Figure 5.1: Derivative of the cost with respect to the threshold for saving 3 agents ($\partial C/\partial t_3$). The blue area represents the points (t_2, t_3) for which this derivative is negative, and the red/yellow area represents the positive region. The green star represents the global optimal solution of $(t_2^*, t_3^*) = (\frac{1}{6}, \frac{1}{12})$.

values. We illustrate this in Figure 5.1, where the black curve shows the optimal value of t_3 for any arbitrary value of t_2. Specifically, the figure shows that if we first optimize over t_3, we get a value different than 1/12 (unless t_2 is set to a high enough value).

3. Figure 5.1 also shows the derivative of the objective function with respect to t_3 for any pair (t_2, t_3). We show in Section 5.5 that as long as $t_2 > t_3$ (under the diagonal line), the sign of the derivative is independent of the actual value of t_2. This may suggest that as long as $t_2 > \frac{1}{12}$, the optimal t_3 should be 1/12. However, this is not the case. In fact, for $t_2 = \frac{1}{12} + \epsilon$ we see that the optimal t_3 is even higher that t_2. The situation becomes even more chaotic if we add one more agent, as the optimal value of t_3

could depend on t_4 as well. However, we show that fortunately none of these complicating behaviors happen, once we set the earlier thresholds to their optimal values. More precisely, once we are optimizing over t_i after setting earlier thresholds to their optimal values, the resulting value we get for t_i would be smaller than all previous values, and it will not depend on the remaining thresholds.

4. Another point worth noting from this example is that the optimal values of t_2 and t_3 would remain the same if we added one or more agents to the initial pool of participants. This is not specific to the uniform distribution, as we prove it more generally in Lemma 5.5.1. We discuss the similarity of this independence to regular auctions next.

Similarities to Myersonian approach. Our main result is the characterization of optimal thresholds for our procurement auction setting and an algorithm to efficiently compute them. One important corollary of our analysis is that the optimal threshold mechanism is independent of the initial number of agents participating in the auction. More precisely, it turns out that the optimal threshold for saving k agents is the value where the marginal contribution to the cost of the current day (virtual cost w.r.t. F) is equal to the savings of having the k-th agent present in the future rounds. Note that the virtual cost function $(x + \frac{F(x)}{f(x)})$ is solely determined by the underlying distribution function F. Also, the marginal gain of having the k-th agent in the future round(s) does not depend on the pool of agents we start with. This is

very similar to Myerson's result for revenue maximization in regular auctions: for n symmetric buyers, the optimal auction is a second-price auction with a reserve price. The reserve price is obtained by setting the virtual valuation (defined as $v - \frac{1-F(v)}{f(v)}$) to zero, and is again independent of the number of agents (n). This similarity is surprising, since in procurement auctions we have to always meet the demand and we cannot use price as a tool to trade off utility across different types. However, using thresholds for saving a different number of agents, we are able to trade off utility across different types and different rounds.

Another similarity is in how we achieve optimality. In revenue maximization, the reserve price is equivalent to *supply reduction*, meaning that depending on the bids, the seller has the right to not sell the item. Note that this means that the optimal auction is not efficient, as the seller will sometimes withhold the object even though the highest bidder has a strictly positive value. For our cost minimization problem, we achieve optimality via what is effectively a *demand increase*. The efficient outcome for each stage is to assign the service to the agent with the lowest overhead cost since every agent can satisfy the demand. However, the mechanism may allocate the production of the service to multiple agents in the hope of decreasing the future costs.[2]

[2]Our solution keeps the total production the same, splits the allocation of the service equally and increases the payment rate accordingly. This is equivalent to increasing the total demand without inflating the payment rate. See Section 5.3 for more details.

Summary of results. The results in this chapter can be summarized as follows:

(a) We model a two-round dynamic procurement auction with abandonment, where the agents leave the auction if they do not meet their overhead costs in a given round. We focus on threshold mechanisms, as they are widely used in practice, and show that they are ex-post incentive compatible for our dynamic auction model. The thresholds are similar to setting reserves for revenue maximization in regular auctions.

(b) Next, we study the optimization problem for finding the optimal set of thresholds. We show that if the distribution F for overhead costs is regular (as defined later), the optimal thresholds are independent of the number of agents participating in the auction. In other words, we do not need to know the number of agents to determine the optimal set of thresholds.

(c) We prove that if the underlying distribution F satisfies certain properties, the optimal thresholds will be monotone, meaning that the optimal threshold for saving i agents is lower than the optimal threshold for saving j agents for any $i > j$. Moreover, we show that this monotonicity helps divide the optimization problem into n separate problems, which ultimately leads to an efficient algorithm to calculate the optimal thresholds in parallel.

5.2 Related Work

Single-parameter mechanism design has been extensively studied in theoretical computer science over the last decade and lead to several interesting results in the intersection of approximation and mechanism design (e.g. [141] and references therein). Over the last few years there has been an increased interest in dynamic mechanism design and specifically, revenue maximization in repeated auctions [142, 143]. The challenge in this line of work has been that depending on the assumptions about when the agents obtain their information, these models become multi-dimensional, leading to a notoriously hard problem in mechanism design (see [144] for a survey).

For example, Ashlagi *et al.* [142] study incentive compatible mechanisms for revenue maximization. In contrast to prior economic literature they require that the mechanism is strongly individually rational, namely the utility of each agent should be non-negative at any stage of the game. One interpretation of strong individual rationality in the context of a dynamic auction is that agents would abandon the service if they ever receive negative utility. Our model of abandonment in a procurement auction setting can be thought of as a relaxation of individual rationality, where each agent expects to achieve a specific level of utility and if she does not meet her target then she abandons the platform.

Different models of dynamic procurement auctions have been studied in the past. The common aspect between these different models is an intertemporal dependency, either on the procurer/buyer side or the suppliers/bidders,

that ties the outcomes of the individual auctions. Examples of such dependencies include:

Capacity constraint: When the bidders are capacity-constrained, their costs increase if they win the current auction (due to higher future capacity utilization). Therefore, capacity-constrained firms face an intertemporal trade-off in sequential auctions: higher profits in the current period lead to lower profits in future periods. This model has been studied over both a finite [126] and an infinite horizon [127].

Switching cost: When a procurer buys goods from competing suppliers repeatedly over time, she may incur an additional switching cost each time she switches from one supplier to another. These costs arise because the buyer must acquire skill at using a new supplier's product, and affect the competition between the incumbent supplier and his rivals [128, 129].

Backlog/holding cost: In dynamic inventory control models, the procurer becomes a retailer who has to repeatedly run a procurement auction among a number of potential suppliers before observing the actual demand. At the end of each period, any unsatisfied demand will be backlogged with a backlog cost and any unsold inventory will be carried over to the next period with a holding cost [130, 131].

Learning through experience: In many industries learning by doing or learning through production experience enables suppliers to provide better

service at lower costs. Lewis and Yildirim [132] consider such model in which the cost of each supplier at each round consists of a (public) intrinsic cost of production, which decreases every time that producer supplies the procurer, and a (private) transitory cost drawn according to a prior distribution. They study how buyer optimally manages dynamic competition among rival suppliers to exploit learning economies.

Piecewise procurement: Sometimes sequential procurement auctions belong to a large-scale project whose subprojects have to be procured in a predetermined order. The project yields its full value once it is completed. The question is then how the procurer optimally designs a procurement auction for each subproject, especially when she cannot write long-term contracts [134, 135].

In comparison to these previous models, we introduce the notion of **abandonment** to the procurement auction, meaning that the suppliers may leave the auction if their received payments do not cover their internal costs. Under this model, it is no longer true that repeating a single-round-optimal auction will lead to assigning the demand to the best set of agents at the best price [136]. To the best of our knowledge, this fundamental model has not been studied in the literature.

5.3 Preliminaries

There are 2 periods and a set of agents N, where $|N| = n$. Each period the mechanism designer wants to allocate a unit of production to a subset of agents. In period $j = 1, 2$, each agent i is characterized by her overhead cost M_i^j and her production cost c_i^j. We assume that the overhead costs are private and independently identically distributed according to a distribution F (independent across both agents and rounds). We will assume that F is a continuous distribution supported on $[0, 1]$.

Let x_i^j be the production percentage allocated to agent i in round j and p^j the anonymous payment rate for round j. The utility of agent i in round j is given by:

$$u_i^j(M_1^j, M_2^j, \cdots, M_n^j) = \begin{cases} x_i^j(p^j - c_i^j) & x_i^j(p^j - c_i^j) \geq M_i^j, \\ -\infty, & x_i^j(p^j - c_i^j) < M_i^j. \end{cases} \tag{5.1}$$

Agent i seeks to maximize her aggregate utility $u_i^1 + u_i^2$ from both rounds. The utility function is capturing the fact that if an agent does not meet her overhead cost M_i^j in round j, she goes out of business and loses everything she gained today. In addition we assume that if an agent receives $-\infty$ utility she will abandon the auction.

We further focus on the case where the individual production costs c_i^j are known to the designer and homogeneous across the agents. For simplicity all our results will assume $c_i^j = 0$ for all i and j but, as we show in Section 5.6, this can be generalized if they are the same for all agents in a particular round

but not necessarily 0, and can vary across rounds. Hence, without loss of generality, the utility of agent i becomes:

$$u_i^j(M_1^j, M_2^j, \cdots, M_n^j) = \begin{cases} x_i^j \cdot p^j & x_i^j \cdot p^j \geq M_i^j, \\ -\infty, & x_i^j \cdot p^j < M_i^j. \end{cases} \tag{5.2}$$

The mechanism designer does not know the overhead costs, M_i^j, which are all identically and independently distributed according to a distribution F, i.e., $M_i^j \sim F$ independent across rounds $j = 1, 2$ and across agents $i = 1, \ldots, n$.

Mechanism. Each agent reports her current overhead cost M_i^j to the designer during round j and the designer decides on the allocation $x_i^j(M_1^j, \cdots, M_n^j)$ for all $i \in N$ and the anonymous payment rate $p^j(M_1^j, \cdots, M_n^j)$. We seek to design a mechanism that minimizes the expected total cost of the outcome

$$\mathbb{E}_{M_i^j \sim F}\left[p^1(M_1^1, \ldots, M_n^1) + p^2(\hat{M}_1^2, \ldots, \hat{M}_n^2)\right],$$

where $\hat{M}_i^2 = M_i^2$ if $x_i^1 \cdot p^1 \geq M_i^1$ and $\hat{M}_i^2 = \infty$ otherwise.

Truthfulness. There are several generalizations of truthfulness once we depart from the standard single-shot environment. Ex-post incentive compatibility requires that agents want to report truthfully their overhead costs if this maximizes their aggregate utility even if they have access to the realization of their overhead costs in advance. For example, in our setting with two rounds, agent i should not have an incentive to report a different value than M_i^1 in

round 1 despite knowing the value M_i^2. Periodic ex-post incentive compatibility relaxes this condition to agents having access to the history of the game and having only distributional assumptions for their future overhead costs. Nevertheless, the simple class of threshold mechanisms that we analyze in this paper satisfies the stronger notion of ex-post incentive compatibility. Each round j is characterized by a choice of n different thresholds $(t_1^j, t_2^j, \ldots, t_n^j)$, where t_i^j represents the maximum amount that the mechanism is willing to pay to save the i-th agent in round j. This is more precisely described in the following definition.

Definition 5.3.1. A single threshold mechanism using thresholds $t_1, \ldots, t_n \in [0, 1]$ is defined as follows: Assume $M_1 < M_2 < \cdots < M_n$ and let us define the predicate $T_k(M_1, \ldots, M_n) = 1$ if and only if $M_k \leq t_k$, in other words the k^{th} smallest value is less than the k^{th} threshold.[3] Let k be the highest index such that $T_k = 1$. Then the mechanism allocation is:

$$x_i = \begin{cases} 1/k & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

and the payment to agent i is $x_i \cdot p$, where p is the total mechanism payment (also the per unit cost of providing the demand) defined as $p = k \cdot \min\{t_k, M_{k+1}\}$. In other words, the cheapest k agents equally provide the service, while each receiving a payment of $\min\{t_k, M_{k+1}\}$.

[3] In the case of ties we need to slightly adjust the description of the mechanism. For the sake of clarity we present the more general version of the mechanism in Appendix C.2 and prove that it is truthful. Since we assume continuous distributions, we can assume no ties for optimizing our objective, without loss of generality.

The mechanism uses the thresholds to determine the number of agents it wishes to allocate the service to. Note that allocating the service to more than one agent is inefficient. Allocating to multiple agents and respecting their overhead costs means that for every agent such that $x_i > 0$ it must be that the agent payment is at least her overhead cost, $x_i \cdot p \geq M_i$.

Proposition 5.3.1. *Any threshold mechanism is truthful*[4] *in the corresponding single-shot game and each agent that has non-zero allocation has non-negative utility.*

Proof. If an agent i is not allocated the service, she receives utility of $-\infty$. Bidding a lower overhead cost may result in her being allocated some part of the demand. There are two scenarios in which this may happen: (1) If there exists some k such that T_k is the highest true predicate both before and after agent i lowered her bid. In this case, it must be that her lower bid is less than or equal to $M_k < M_i$. This results in a payment equal to M_k, which makes her utility $-\infty$ again. (2) If T_k is not the highest true predicate after agent i lowers her bid. Assume that the new highest predicate satisfied is T_w for some $w > k$. Since T_w was not true before, it must be that the threshold t_w is now the critical value, therefore each agent receives a payment equal to t_w. But since T_w was false before, we know that $t_w < M_i$, meaning that agent i will receive $-\infty$ utility.

[4] Assuming no ties.

If agent i is allocated the service, notice that her payment is independent of her actual overhead cost. Reporting a lower overhead cost does not change her allocation nor payment. Similarly, if she reports a higher amount, she will receive the same payment, as long as she is still being allocated the service. If her increase makes her not being allocated, then her utility becomes $-\infty$. In neither case is deviating from reporting the true overhead cost profitable. □

We now define a threshold mechanism for a two-round game.

Definition 5.3.2. A threshold mechanism for a two round game is characterized by two sets of thresholds $\mathbf{t^1} = (t_1^1, \ldots, t_n^1)$ and $\mathbf{t^2} = (t_1^2, \ldots, t_n^2)$. For any round j, we allocate the demand to at least i agents, if there are i bids below t_i^j.

While technically the threshold mechanism defined in the second round could depend on the number of surviving agents, the optimal mechanism in the last round is oblivious to this fact; it will always allocate the service to a single agent and offer her a payment equal to the second lowest bid or the top of the distributional support if only one agent has survived. Since the mechanism is only feasible if it always allocates the entire demand, we need to have that $t_1^j = 1$ (the upper bound of the support of F) and therefore we will be omitting t_1 from now on.

Proposition 5.3.2. *A threshold mechanism for a dynamic game is ex-post incentive compatible.*

Proof. It is easy to see that for $j = 2$ (the last round), truthfulness of the threshold mechanism in the single-shot version implies that reporting the truth in the last round is optimal for each agent. For $j = 1$, we have to argue that deviating from the truth does not increase the aggregate utility for the agent. Since the mechanism is independent of the outcome of round 1, the only way that the reported overhead cost in round 1 affects the second round is if the agent is not allocated in the first round, hence has to abandon the auction. Instead, the agent could misreport a smaller overhead cost in order to ensure some allocation in round 1 so as to be considered in round 2. But in this case the aggregate utility of this agent remains $-\infty$, hence she cannot benefit from the deviation. □

As mentioned earlier, our objective in designing a threshold mechanism is to minimize the total payment of our allocation. In other words, we seek a mechanism (x, p) with thresholds $(\mathbf{t^1}, \mathbf{t^2})$ such that it minimizes the total payment. The optimal mechanism for the second round is independent of what happens during the first round and there is no reason to allocate the production of the service to more than one agent.

Proposition 5.3.3. *The optimal threshold mechanism for the second round of a two-round auction is always equal to* $t_2^2 = t_3^2 = \cdots = t_n^2 = 0$.

Proof. Setting $t_2^2 > 0$ means that with some probability two agents will be allocated the service resulting in a payment more than the second lowest bid. Note that allocating the service to the second lowest agent does not result

in any benefit in the future (since this is the last round). On the contrary, setting all thresholds for round two to 0 ensures that we allocate the service to the agent with the lowest bid, and the payment would be equal to the second lowest bid. Similarly, setting any t_i^2 to a non-zero value is a sub-optimal choice. Therefore, the optimal mechanism in round 2 is to set all thresholds t_i^2 to zero for $i \geq 2$. (As always we have $t_1^2 = 1$.) □

Note that when the threshold mechanism allocates to an agent, it ensures that the payment she receive is at least her reported overhead cost so she will not abandon the auction. This is not necessarily needed for the second round according to the definition of our objective. If we allow the mechanism to allocate to an agent and not respect her overhead cost, then we could simply add an additional round. In that case, any feasible mechanism must ensure that one agent survives to the third round; therefore, the payments should satisfy her overhead cost in the second round as well. Thus our analysis exactly captures this case when we only focus on the first two rounds.

The main result of our paper is to characterize the first round optimal threshold mechanism for dynamic procurement. It is important to note the connection of our problem to revenue maximization where effectively we use a similar analysis in terms of virtual costs. An alternative way to interpret our mechanism is that it implements a form of supply increase to reduce the aggregate cost of the mechanism. Our results hold for natural assumptions on the distribution of the overhead cost defined below.

Definition 5.3.3 (Regularity [145])**.** We say that a probability distribution f (with cumulative distribution function F) supported on $[0, 1]$ is regular if its virtual cost function defined as $x + \frac{F(x)}{f(x)}$ is monotone increasing.

Definition 5.3.4 (Order statistics [146])**.** Let $X_1, ..., X_n$ be a random sample of size n (independent) from a distribution F and $X_{1:n} \leq X_{2:n} \leq ... \leq X_{n:n}$ be the order statistics obtained by arranging X_i's in non-decreasing order. We denote by $\mu_{r:n}$ the expectation of the r^{th} order statistic, i.e.:

$$\mu_{r:n} = \mathbb{E}[X_{r:n}]$$

Definition 5.3.5 (Diminishing returns of order statistics)**.** We say that the r^{th} order statistic of a distribution F has the diminishing returns property if

$$\mu_{r:n-1} - \mu_{r:n} \geq \mu_{r:n} - \mu_{r:n+1}, \qquad \forall n > r.$$

Our main theorem is stated below. A surprising property we find is that the mechanism does not need to know the initial number of agents that participate in any round of the auction.

Theorem 5.3.4. *If distribution F satisfies the regularity condition (Definition 5.3.3), and its second order statistic has the diminishing returns property (Definition 5.3.5), then the optimal threshold mechanism can be found in polynomial time.*

The remainder of this chapter is organized as follows. In Section 5.4 we define the canonical threshold mechanism and provide a few examples.

In Section 5.5 we present our main theorem proving the optimality of the canonical threshold mechanism. In Section 5.6 we discuss generalizations of the model where our results still hold. Finally, in Section 5.7 we discuss significant departures from our setting via breaking various types of homogeneity and symmetry and propose future directions.

5.4 Mechanism

Our objective in designing a threshold mechanism is to minimize the payment of our allocation. We will use $C_n(t_2, ..., t_n)$ to denote the aggregate cost of two rounds given a specific set of thresholds $(t_2, \ldots, t_n)$ for the first round, where n is the number of agents in round 1.

Example 1 (2 agents, 2 rounds)**.** To develop intuition for the general problem, we first start with a simple example. Suppose we have 2 agents in round 1. We want to answer the following questions: "When is it beneficial to keep both agents alive for the second round? Is there a more efficient way than committing to save a particular number of agents a-priori?"

Since we have only two agents and one is always picked, we only need one threshold denoted by $t \in [0, 1]$ to determine whether or not picking the second agent is beneficial. If both bids are below t, we pick both agents and pay them t each; otherwise, we pick the cheaper agent and the payment would be equal to the higher bid. To find the optimal t, we have to calculate the expected cost as a function of t. We assume the overhead costs are drawn according to a uniform distribution supported on $[0, 1]$, i.e., $M_1, M_2 \sim U[0, 1]$.

Table 5.1: Different scenarios of the bids of two agents (Example 1) with respect to the threshold t. Here green represents which agents survive to the next round, and red represents the payment to each of those green agents.

#	Case	Probability	Expected Cost	Picture
1	$M_1, M_2 \leq t$	t^2	$2t+\frac{2}{3}$	1 t M_2 M_1 0
2	$M_1 \leq t < M_2$ or $M_2 \leq t < M_1$	$2t(1-t)$	$t+\frac{1}{2}(1-t)+1$	1 M_2 t M_1 0
3	$M_1, M_2 > t$	$(1-t)^2$	$t+\frac{2}{3}(1-t)+1$	1 M_2 M_1 t 0

Three different cases can happen regarding the bids (in round one) as shown in Table 5.1.

For example, in the first case when both bids are below t, we decide to keep both agents alive; therefore, we have to pay t to each of them. There is also a cost of $\mu_{2:2} = 2/3$ which is the expected cost of the next round, given that both agents will be available and we have to pay the highest bid (out of those two uniform $[0, 1]$ bids) to the lowest bidder. On the other hand, when we keep one agent alive in cases 2 and 3, that single agent faces no competition in the next round and will bid $M_i = 1$ (which we will have to accept). Note that when we save only one agent, the payment is determined by the second bid, since we always assume $t_1 = 1$.

Multiplying the costs by their corresponding probabilities and adding up cases we get:

$$\mathbb{E}[cost] = t^2\left(2t + \frac{2}{3}\right) + 2t(1-t)\left(\frac{1}{2} + \frac{t}{2} + 1\right) + (1-t)^2\left(\frac{2}{3} + \frac{t}{3} + 1\right).$$

Optimizing over the threshold value t, we get $t^* = \frac{1}{6}$, which evaluates to a cost of $\frac{539}{324} \approx 1.6636$.

Example 2 (3 agents, 2 rounds)**.** In this example we want to answer the following questions: "When is it beneficial to keep all three agents alive for the second round? Also, how does the threshold for picking 2 agents change compared to the previous example with only 2 agents?"

Let t and t' be the thresholds for picking 2 and 3 agents in round one, respectively. Doing the same calculations as in the previous example, we get the following expected cost function: (we assume that $t' \leq t$, but we can also calculate the expected cost for the case of $t' > t$ and check that the optimal solution is indeed in the $t' \leq t$ region.)

$$\begin{aligned}\mathbb{E}[cost] =& t'^3\left(3t' + \frac{1}{2}\right) + 3t'^2(t-t')\left(2(t' + \frac{t-t'}{2}) + \frac{2}{3}\right) \\ &+ 3t'(t-t')^2\left(2(t' + \frac{2(t-t')}{3}) + \frac{2}{3}\right) \\ &+ (t-t')^3\left(2(t' + \frac{3(t-t')}{4}) + \frac{2}{3}\right) + 3t^2(1-t)\left(2t + \frac{2}{3}\right) \\ &+ 3t(1-t)^2\left(\frac{1+2t}{3} + 1\right) + (1-t)^3\left(\frac{1+t}{2} + 1\right). \end{aligned} \tag{5.4}$$

Optimizing over threshold values t, t' we get $(t, t') = (\frac{1}{6}, \frac{1}{12})$ which evaluates to a cost of 1.49149.

Observe that in the previous 2 examples, the threshold of saving 2 agents was $\frac{1}{6}$, regardless of whether we started with 2 or 3 agents. In addition, the expected cost (5.4) has the property that for any t', the optimal value of t is 1/6. Also, for any value of $t \geq \frac{1}{6}$, the cost is minimized at $t' = \frac{1}{12}$. These observations lead us to the idea that the optimal thresholds can be calculated individually, through the following notion of canonical thresholds.

Definition 5.4.1 (Canonical thresholds)**.** The canonical threshold for saving i agents, denoted by $\hat{t}_i$, is the optimal value for t_i when all previous thresholds are set to one, and all remaining thresholds are set to zero. More precisely,

$$\begin{aligned} \hat{t}_i \equiv \quad & \underset{t_i}{\text{argmin}} \quad C_n(t_2, ..., t_n) \\ & \text{s.t.} \qquad t_2 = \cdots = t_{i-1} = 1, \\ & \qquad\quad t_{i+1} = \cdots = t_n = 0. \end{aligned} \tag{5.5}$$

In Section 5.5 we show that the canonical thresholds defined above are indeed optimal thresholds for minimizing the objective function $C_n(t_2, ..., t_n)$. To prepare the ground for this result, we first establish some properties of our objective function. In particular, in Theorem 5.4.1, we calculate the partial derivative of the objective function with respect to any threshold t_i. For this theorem, we have to define the following notation.

Notation. Recall that we defined the predicate $T_k(M_1, ..., M_n) = 1$ if there are at least k bids below t_k. We define the vector $\mathbf{M} = (M_1, ..., M_n)$ to be the vector of all private values and we write $T_k(\mathbf{M}) = 1$, or for short $T_k = 1$, if the k^{th} predicate is satisfied. Otherwise, we write $T_k = 0$.

Given that the first i bids are below t_i (hence predicate i is satisfied), we define $P_{i,n}$ as the probability that the remaining bids are above t_i so as not to satisfy any higher predicate $(T_{i+1}, ..., T_n)$. More precisely, we define

$$P_{i,n} = \Pr\left[M_{i+1}, ..., M_n > t_i, T_{i+1} = ... = T_n = 0 \mid M_1, ..., M_i \leq t_i\right], \quad (5.6)$$

where we assume $P_{n,n} = 1$ (since higher bids/thresholds do not exist for this case). An important property that we use in our proofs is that by this definition, $P_{i,n}$ is independent of all lower thresholds $(t_2, ..., t_{i-1})$.

Finally, given a vector of all private values $\mathbf{M}$, we define $g(\mathbf{M}, t_2, ..., t_n)$ to be the total cost of the mechanism using thresholds $t_2, ..., t_n$. This total cost consists of a deterministic cost for the current round (since the bids are given by $\mathbf{M}$) and an expected cost for the future round(s). With our earlier notation, $C_n(t_2, ..., t_n) = \mathbb{E}_{\mathbf{M}}[g(\mathbf{M}, t_2, ..., t_n)]$. We are now ready to calculate the partial derivative of the objective function.

Theorem 5.4.1. *The derivative of the cost with respect to any threshold t_i is given by:*

$$\frac{\partial C_n(t_2, ..., t_n)}{\partial t_i} = i\binom{n}{i} P_{i,n} F(t_i)^i + i\binom{n}{i} P_{i,n} F(t_i)^{i-1} f(t_i) \times \quad (5.7)$$
$$\mathbb{E}_{\mathbf{M}}\Big[i \times t_i + \mu_{2:i} - g(\mathbf{M}, t_2, ..., t_n) \mid T_i = ... = T_n = 0, t_i < M_{i:n} < t_i + \epsilon\Big],$$

where $\epsilon \to 0$.[5]

[5]To be precise, we need to put $\lim_{\epsilon \to 0}$ in the right-hand side of equation (5.7), but we omit the limit for brevity, keeping in mind that ϵ is a small enough constant.

We differ the full proof to Appendix C.5. Here we provide some intuition on different parts of this expression. Roughly speaking, the cost $C_n(t_2, ..., t_n)$ is determined by the "active" threshold, which corresponds to the highest predicate that is satisfied. As long as we do not change the active threshold, perturbing the remaining thresholds should not change the cost, therefore the derivative should be zero with respect to them. When we think of the derivative with respect to a particular t_i, we want to know how much the cost would increase/decrease if we change t_i to $t_i + \epsilon$. There are two scenarios where this perturbation changes the cost:

1. The first scenario is when there are exactly i agents below t_i. This corresponds to $\binom{n}{i}F(t_i)^i$ in (5.7). We also want the remaining bids to be above t_i in a way that higher predicates are not satisfied (so that t_i is active), which is captured by $P_{i,n}$. Finally in this case, when we add ϵ to t_i, all those i agents receive ϵ more payment, which corresponds to the multiplicative term i in (5.7).

2. The second scenario is when t_i becomes active after we add ϵ to it. This requires that the i^{th} bid is between t_i and $t_i + \epsilon$, which is why we get $i\binom{n}{i}F(t_i)^{i-1}f(t_i)$. We again need the remaining bids to be above $t_i + \epsilon$ and to not satisfy any higher predicate ($P_{i,n}$). The change in the cost is more complicated in this scenario. We know that at $t_i + \epsilon$ we are going to save i agents and therefore the cost would be roughly $i \times t_i$ for this round, and $\mu_{2:i}$ for the next round. However, it is not clear how

many agents we were saving at t_i. That is why we have the expectation of the cost with negative sign, while the expectation is conditioned to this particular scenario in which there are exactly $i-1$ agents with bids below t_i.

5.5 Optimality of Canonical Thresholds

In this section, we study the optimality of the canonical thresholds. We first begin by showing that canonical thresholds form a monotone decreasing sequence. While this property seems intuitive, it is not necessarily true if the underlying distribution F does not satisfy our two assumptions of regularity and diminishing returns property of the second-order statistic, discussed in Section 5.3. See Appendix C.4 for an example of an irregular distribution F, for which the optimal thresholds are non-monotone.

Lemma 5.5.1. *For a regular distribution F that its second order statistic has the diminishing returns property (Definition 5.3.5), the canonical thresholds are monotone non-increasing and independent of the number of agents n.*

Proof. By definition, $\hat{t}_i$ is the optimal value for t_i when $t_2 = \cdots = t_{i-1} = 1$ and $t_{i+1} = \cdots = t_n = 0$. To find the optimal t_i, we start from the general expression (5.7) for the derivative and show that it simplifies as follows whenever $t_i \leq t_{i-1}$ (which is true here since $t_{i-1} = 1$). When the i^{th} bid is between t_i and $t_i + \epsilon$, $T_i = ... = T_n = 0$, and $t_i \leq t_{i-1}$, we would save $i-1$ agents and therefore

$g(\mathbf{M}, t_2, ..., t_n) = (i-1)t_i + \mu_{2:i-1}$. This simplifies equation (5.7) to:[6]

$$\begin{aligned}\frac{\partial C_n(t_2, ..., t_n)}{\partial t_i} &= i\binom{n}{i} P_{i,n} \times \left[F(t_i)^i + F(t_i)^{i-1} f(t_i)\big[t_i + \mu_{2:i} - \mu_{2:i-1}\big]\right] \\ &= i\binom{n}{i} P_{i,n} F(t_i)^{i-1} f(t_i) \left[t_i + \frac{F(t_i)}{f(t_i)} + \mu_{2:i} - \mu_{2:i-1}\right], \forall t_i \leq t_{i-1}.\end{aligned} \tag{5.8}$$

Other than the trivial roots $t_i = 0$ and $t_i = 1$ (which are local maximizers), there is a single root for this derivative that determines $\hat{t}_i$ as follows:

$$\hat{t}_i + \frac{F(\hat{t}_i)}{f(\hat{t}_i)} = \mu_{2:i-1} - \mu_{2:i}. \tag{5.9}$$

Therefore, we have $\hat{t}_i \geq \hat{t}_j$ for all $i \leq j$, since the left-hand side is a monotone increasing function and the right-hand side is a constant, monotone non-increasing in i. Also note that (5.9) makes $\hat{t}_i$ independent of n (as long as $n \geq i$). This concludes the proof. □

Now, we prove that the canonical thresholds provide the global optimal solution for minimizing the expected cost of the auction. In Lemma 5.5.2 we show that when the previous thresholds are set to 1, as we increase threshold t_i from zero to its canonical value $\hat{t}_i$, the expected cost $C_n(t_2, ..., t_n)$ decreases; and as we increase t_i beyond $\hat{t}_i$, the cost increases again.

Lemma 5.5.2. *If $t_k = 1$ for all $k \leq i-1$, then $\frac{\partial}{\partial t_i} C(t_2, ..., t_n)$ is non-positive for $t_i \in (0, \hat{t}_i)$, zero at $t_i = \hat{t}_i$, and non-negative for $t_i \in (\hat{t}_i, 1)$.*[7]

[6]For consistency of notation, we define $\mu_{2:1} = 1$. This is because when we save i agents in round one, the expected cost of the second round would be $\mu_{2:i}$ for $i \geq 2$, and 1 if $i = 1$.

[7]Note that whenever $t_{i-1} = 1$, the previous thresholds $t_2, ..., t_{i-2}$ are irrelevant. Therefore, this lemma holds even if we only had $t_{i-1} = 1$. However, we state the lemma as is for the sake of the next lemma.

Proof. Since $t_{i-1} = 1$, we can again use the simplified version of the derivative (5.8) instead of the general version (5.7) for all t_i. Since $P_{i,n}$, $F(t_i)$, and $f(t_i)$ are all non-negative, we have to show that $t_i + \frac{F(t_i)}{f(t_i)} + \mu_{2:i} - \mu_{2:i-1}$ is non-positive for $t_i \in (0, \hat{t}_i)$, zero at $t_i = \hat{t}_i$, and non-negative for $t_i \in (\hat{t}_i, 1)$. Assuming that the virtual cost is monotone non-decreasing, it suffices to show that $t_i + \frac{F(t_i)}{f(t_i)} + \mu_{2:i} - \mu_{2:i-1} = 0$ at $t_i = \hat{t}_i$, which is true due to (5.9). (Note that $t_i + \frac{F(t_i)}{f(t_i)} + \mu_{2:i} - \mu_{2:i-1}$ is strictly negative/positive at 0/1, therefore $\hat{t}_i$ is a fractional point.) □

The previous lemma shows that the canonical threshold $\hat{t}_i$ is the global minimizer of the cost when $t_2 = ... = t_{i-1} = 1$, independent of the values of the remaining thresholds $t_{i+1}, ..., t_n$. However, in the following lemma and its corollary, we show that this holds even if we lower the value of the previous thresholds from 1 to their canonical values.

Lemma 5.5.3. *If $t_k = \hat{t}_k$ for all $k \leq i-1$, then $\frac{\partial}{\partial t_i} C(t_2, ..., t_n)$ is non-positive for $t_i \in (0, \hat{t}_i)$, zero at $t_i = \hat{t}_i$, and non-negative for $t_i \in (\hat{t}_i, 1)$.*

Proof. Note that compared to the previous lemma, we only lowered the value of $t_2, ..., t_{i-1}$ from 1 to their canonical value $t_k = \hat{t}_k$. One can argue from (5.7) that this lowering of thresholds does not change the derivative for any $t_i \in [0, \hat{t}_{i-1}]$. This is true because we can use equation (5.8) in this region, which shows that the derivative is independent of $t_2, ..., t_{i-1}$, whenever $t_i \leq t_{i-1}$ (remember that $P_{i,n}$ is independent of $t_2, ..., t_{i-1}$). Figure 5.2 shows an example of how lowering the thresholds $t_2, ..., t_{i-1}$ affects the derivative with respect to t_i.

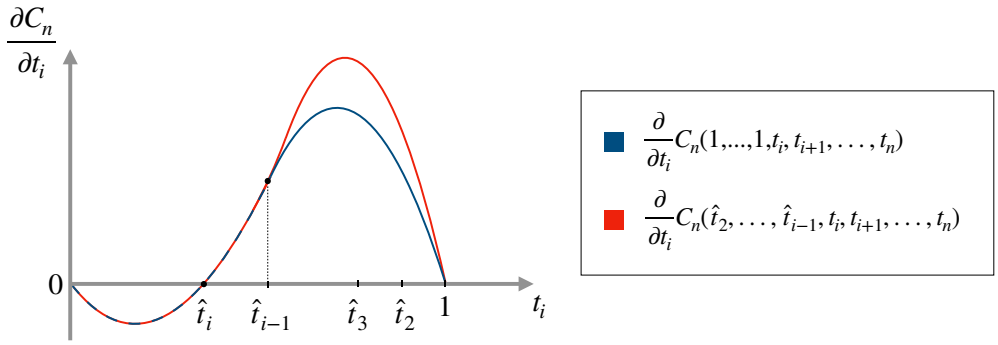

Figure 5.2: Derivative of the cost with respect to t_i when: (blue) the previous thresholds are set to one, (red) the previous thresholds are lowered to their canonical values.

Note that from Lemma 5.5.1 we know that $\hat{t}_i \leq \hat{t}_{i-1}$. This immediately implies that $\frac{\partial}{\partial t_i} C(t_2, ..., t_n)$ is non-positive for $t_i \in (0, \hat{t}_i)$, zero at $t_i = \hat{t}_i$, and non-negative for $t_i \in (\hat{t}_i, \hat{t}_{i-1})$. Therefore, we only need to show that the derivative is non-negative for $t_i \geq \hat{t}_{i-1}$. To do this, we show that the lowering of thresholds $t_2, ..., t_{i-1}$ indeed increases the derivative in this region, i.e.,

$$\frac{\partial}{\partial t_i} C(\hat{t}_2, ..., \hat{t}_{i-1}, t_i, t_{i+1}, ..., t_n) \geq \frac{\partial}{\partial t_i} C(1, ..., 1, t_i, t_{i+1}, ..., t_n), \quad \forall t_i \geq \hat{t}_{i-1} \tag{5.10}$$

which implies the non-negativity of the derivative, since the right hand side is non-negative due to Lemma 5.5.2. To prove (5.10), note that from (5.7), comparing the above two derivatives is equivalent to showing that

$$\mathbb{E}_{\mathbf{M}}\big[g(\mathbf{M}, \hat{t}_2, ..., \hat{t}_{i-1}, t_i, t_{i+1}, ..., t_n) \mid A\big] \leq \mathbb{E}_{\mathbf{M}}\big[g(\mathbf{M}, 1, ..., 1, t_i, t_{i+1}, ..., t_n) \mid A\big],$$

where A is the event that there are exactly $i-1$ bids below t_i and we save at most those $i-1$ agents. Note that this expected cost is exactly equal to the

situation if we had only $i-1$ agents in the auction, and we knew that their bids are upper bounded by t_i. In other words, it suffices to show that

$$\tilde{C}_{i-1}(\hat{t}_2, ..., \hat{t}_{i-1}) \leq \tilde{C}_{i-1}(1, ..., 1), \tag{5.11}$$

where $\tilde{C}_{i-1}$ is the expected cost in a game with $i-1$ agents with distribution $\tilde{F}$ which is obtained from truncating F to have the support $[0, t_i]$ (note that distribution $\tilde{F}$ only applies to the first day, and on day 2 the bids are again drawn according to the original distribution F).

To show (5.11), we use the following set of inequalities:

$$\tilde{C}_{i-1}(\hat{t}_2, \hat{t}_3, \hat{t}_4, ..., \hat{t}_{i-1}) \leq \tilde{C}_{i-1}(1, \hat{t}_3, \hat{t}_4, ..., \hat{t}_{i-1})$$

$$\tilde{C}_{i-1}(1, \hat{t}_3, \hat{t}_4, ..., \hat{t}_{i-1}) \leq \tilde{C}_{i-1}(1, 1, \hat{t}_4, ..., \hat{t}_{i-1})$$

$$\vdots$$

$$\tilde{C}_{i-1}(1, 1, ..., 1, \hat{t}_{i-1}) \leq \tilde{C}_{i-1}(1, 1, ..., 1, 1)$$

Each of the above inequalities is implied by Lemma 5.5.2, since this lemma says that the derivative with respect to any t_k is non-negative for $t_k \geq \hat{t}_k$, as long as the previous thresholds are all equal to one. Therefore, increasing any t_k from $\hat{t}_k$ to 1 cannot decrease the cost. The only concern here is that the thresholds $\hat{t}_k$ were calculated for the auction with n agents and distribution F, while we are using the same thresholds here for the auction with $i-1$ agents and truncated distribution $\tilde{F}$. The reason why we are allowed to do this is that neither changing the number of agents nor truncating the distribution can

affect the optimality of $\hat{t}_k$. This is because $\hat{t}_k$ is the solution of the following equation:

$$t_k + \frac{F(t_k)}{f(t_k)} + \mu_{2:k} - \mu_{2:k-1} = 0$$

In addition to being independent of n, this equation is invariant to conditioning F from above. In other words, for the truncated distribution $\tilde{F}$ we have:

$$\tilde{F}(t) = \frac{F(t)}{F(t_i)}, \qquad \tilde{f}(t) = \frac{f(t)}{F(t_i)}, \quad \forall t \leq t_i$$

Hence $\frac{\tilde{F}(t)}{\tilde{f}(t)} = \frac{F(t)}{f(t)}$, which implies having $t_k = \hat{t}_k$ (for $k = 2, ..., i-1$) gives a lower cost compared to $t_k = 1$, regardless of having distribution F or $\tilde{F}$.[8] $\square$

Since we showed that the derivative (with respect to t_i) is non-positive up to $\hat{t}_i$ and non-negative afterwards, we arrive at the optimality of $\hat{t}_i$.

Corollary 5.5.4. *If $t_k = \hat{t}_k$ for all $k \leq i-1$, then $C_n(t_2, ..., t_n)$ is minimized at $t_i = \hat{t}_i$, independent of the values of the remaining thresholds $t_{i+1}, ..., t_n$.*

So far we showed that as long as the previous thresholds are set to their canonical values, $\hat{t}_i$ is the global optimal value for t_i. To achieve the global optimal values for the entire set of thresholds $(t_k, k = 2, ..., n)$ it suffices to use the previous lemma in an inductive manner.

[8] This is similar to revenue maximization where if we condition F to be above a certain value v and obtain the conditional distribution $\tilde{F}$, we have that $1 - \tilde{F}(x) = \frac{1-F(x)}{1-F(v)}$ and $\tilde{f}(x) = \frac{f(x)}{1-F(v)}$. This implies that the inverse hazard rate and as a result the virtual value functions of these distributions remain the same.

Theorem 5.5.5. *If distribution F satisfies the regularity condition (Definition 5.3.3), and its second order statistic has the diminishing returns property (Definition 5.3.5), then the global optimal thresholds that minimize $C_n(t_2, ..., t_n)$ are*

$$t_k^* = \hat{t}_k, \quad \forall k.$$

Proof. Let us assume that this is not true and there exists another set of thresholds $(t'_2, ..., t'_n)$ with cost smaller than $C_n(\hat{t}_2, ..., \hat{t}_n)$. Looking at t_2, Corollary 5.5.4 can be used without any condition on the remaining thresholds, which immediately implies that either $t'_2 = \hat{t}_2$, or we can change it to $\hat{t}_2$ without increasing the cost. Given $t'_2 = \hat{t}_2$, we can now use this argument again for t_3 and conclude that $t'_3 = \hat{t}_3$. Repeating this argument, we arrive at $C_n(t'_2, ..., t'_n) = C_n(\hat{t}_2, ..., \hat{t}_n)$, which contradicts our starting assumption. □

5.6 Extensions

Our results extend to several generalizations, as long as the agents remain homogeneous. In particular, the following extensions hold individually or in combination with the others:

Symmetric capacities. Throughout this paper we assumed that each agent is able to meet the entire demand, or equivalently there are no capacities on the agents. However, in practice we can have the constraint that the assignments should satisfy $x_i \leq \overline{x}_i$, where $\overline{x}_i$ is the capacity of agent i. We argue that our

results hold if the agents have the same capacity, i.e., $\overline{x}_i = \overline{x}$ for all i. To see this, let $m = \lceil 1/\overline{x} \rceil$ be the minimum number of agents that we need to keep in all rounds. To guarantee that we meet the demand on day 2, we have to set $t_2 = ... = t_m = 1$ in day 1. For the remaining thresholds t_k ($k > m$), note that if we save k agents from day 1 to day 2, the expected cost on day 2 would be $m \times \mu_{m+1:k}$. Therefore, similarly to (5.9), the optimal value of t_k is where the virtual cost matches the savings of having k agents versus having $k-1$ agents, i.e., $t_k + F(t_k)/f(t_k) = m(\mu_{m+1:k-1} - \mu_{m+1:k})$. Note that in this case we need the $(m+1)$-st order statistics to satisfy the diminishing returns property of Definition 5.3.5.

Changing distributions. Note that the analysis of our threshold algorithm did not use the fact that the distribution in each round was the same. Let F_1 and F_2 be the distributions for the overhead costs for day 1 and 2 respectively. We define the canonical threshold to satisfy

$$\hat{t}_i + \frac{F_1(\hat{t}_i)}{f_1(\hat{t}_i)} = \mu_{2:i-1} - \mu_{2:i}. \tag{5.12}$$

where μ is defined according to the order statistics of F_2. As long as the regularity condition holds for F_1, and F_2 has the diminishing returns property of the second order statistic, our theorem still holds since our proof only requires that $\mu_{2:i-1} - \mu_{2:i}$ is a decreasing function of i.

Non-zero service costs. Assume that the agents incur a cost of c^1 and c^2 for providing a unit of demand on day 1 and 2, respectively (but still have the same cost as other agents each round). In this case, the definition of a threshold mechanism changes slightly compared to Definition 5.3.1. After finding the highest predicate T_k that is satisfied, the first k agents with lowest bids are allocated an assignment of $x_i = 1/k$, while receiving a payment of $p \cdot x_i$, except that the payment now increases to $p = k \cdot \min\{t_k, M_{k+1}\} + c$ (where c is the service cost of the corresponding day). This is required to ensure that for any agent with strictly positive allocation we have $x_i(p - c_i) \geq M_i$.

Now we argue that the canonical thresholds are still optimal for the case of non-zero costs. This is because any feasible allocation will result in an additional cost of $\sum_i x_i \cdot c = c$ to the mechanism designer. As a result, the optimal threshold mechanism for non-zero service cost ($c > 0$) corresponds to designing the optimal mechanism for $c = 0$, and adding the cost c to the payment rate p so as to ensure that the utilities of the agents are the same. This shows that our assumption of $c_i^j = 0$ (for all i, j) throughout the paper was without loss of generality.

5.7 Conclusion

In this chapter we studied a dynamic procurement auction for n symmetric agents. We assumed 3 different properties for the agents that were crucial to achieve the optimality of the canonical thresholds: (i) we assumed a common distribution F for the overhead costs, (ii) we assumed that the

per-unit cost of providing the service is the same for all agents, and (iii) we assumed that each agent can provide the entire demand. Relaxing any of these assumptions breaks the symmetry of the agents and opens a new research question for future work.

If we assume a different distribution F_i for each agent's overhead cost, then the savings from allocating the service to k agents and having them participate in the future rounds depend on the identity of those agents. This could potentially lead to having a different threshold for any subset of agents, which would make the problem computationally intractable. On the other hand, if we assume that agents have different per-unit costs, the optimal assignment would not be trivial, even if the set of agents with non-zero assignments are known. In other words, if we want to save a particular set of k agents, the optimal assignment is not necessarily $1/k$, and it depends on the per-unit costs of those particular k agents. The same challenge holds when we consider different capacities for the agents, as equal assignments of $1/k$ may not even be feasible in that setting.

Appendices

Appendix A

Missing Proof from Chapter 2

Proof of Theorem 2.5.3: The summation of supermodular set functions is supermodular, so we only need to prove the supermodularity for a fixed edge $\{i, j\} \in \mathcal{E}$. We can also drop positive constants like R_{ij}. Define $f_{ij}(A)$ and $f'_{ij}(A)$ as follows:

$$f_{ij}(A) = z^0_{ji} \left(\sum_{k \in \mathcal{N}} z^k_{ij} p_k \right)^2, f'_{ij}(A) = z^0_{ji} \left(\sum_{k \in \mathcal{N}} z^k_{ij} q_k \right)^2$$

We now prove that $f_{ij}(A)$ is supermodular. A similar proof works for $f'_{ij}(A)$. We want to show that:

$$f_{ij}(A \cup \{e\}) - f_{ij}(A) \leq f_{ij}(B \cup \{e\}) - f_{ij}(B), \tag{A.1}$$

for every $A \subseteq B \subseteq \mathcal{E}$, $e \in \mathcal{E}$, and $e \notin B$. For any k, let a^k_{ij} be the change in z^k_{ij} when we add e to A, i.e.:

$$a^k_{ij} = z^k_{ij}(A \cup \{e\}) - z^k_{ij}(A),$$

where $z^k_{ij}(A)$ is just z^k_{ij}, calculated based on the edges in A. Similarly, let b^k_{ij} be defined for B and $B \cup \{e\}$. We have $a^k_{ij} \leq b^k_{ij}$, because any new path in A created by adding e is also a new path in B. Another fact is that $z^k_{ij}(A) \leq z^k_{ij}(B)$, because adding more edges cannot decrease the number of paths between any

pair of vertices (i.e., $f(A)$ is a monotone increasing function). Now we prove (A.1):

$$
\begin{aligned}
& f_{ij}(B \cup \{e\}) - f_{ij}(B) && \text{(A.2)}\\
=& z^0_{ji}(B \cup \{e\}) \left(\sum_{k \in \mathcal{N}} z^k_{ij}(B \cup \{e\}) p_k \right)^2 - z^0_{ji}(B) \left(\sum_{k \in \mathcal{N}} z^k_{ij}(B) p_k \right)^2 && \text{(A.3)}\\
=& (z^0_{ji}(B) + b^0_{ji}) \left(\sum_{k \in \mathcal{N}} b^k_{ij} p_k + \sum_{k \in \mathcal{N}} z^k_{ij}(B) p_k \right)^2 - z^0_{ji}(B) \left(\sum_{k \in \mathcal{N}} z^k_{ij}(B) p_k \right)^2 && \text{(A.4)}\\
\geq& (z^0_{ji}(A) + b^0_{ji}) \left(\sum_{k \in \mathcal{N}} b^k_{ij} p_k + \sum_{k \in \mathcal{N}} z^k_{ij}(B) p_k \right)^2 - z^0_{ji}(A) \left(\sum_{k \in \mathcal{N}} z^k_{ij}(B) p_k \right)^2 && \text{(A.5)}\\
\geq& (z^0_{ji}(A) + b^0_{ji}) \left(\sum_{k \in \mathcal{N}} b^k_{ij} p_k + \sum_{k \in \mathcal{N}} z^k_{ij}(A) p_k \right)^2 - z^0_{ji}(A) \left(\sum_{k \in \mathcal{N}} z^k_{ij}(A) p_k \right)^2 && \text{(A.6)}\\
\geq& (z^0_{ji}(A) + a^0_{ji}) \left(\sum_{k \in \mathcal{N}} a^k_{ij} p_k + \sum_{k \in \mathcal{N}} z^k_{ij}(A) p_k \right)^2 - z^0_{ji}(A) \left(\sum_{k \in \mathcal{N}} z^k_{ij}(A) p_k \right)^2 && \text{(A.7)}\\
=& f_{ij}(A \cup \{e\}) - f_{ij}(A). && \text{(A.8)}
\end{aligned}
$$

In (A.3), (A.4) we just applied the definitions of f_{ij} and b^k_{ij}, respectively. In (A.4), the aggregate coefficient of $z^0_{ji}(B)$ is positive, so using the fact that $z^0_{ji}(B) \geq z^0_{ji}(A)$, we get (A.5). To get (A.6), note that quadratic function $(x+\alpha)^2 - x^2 < (y+\alpha)^2 - y^2$ for $x < y$ and fixed $\alpha > 0$. Setting $\alpha = \sum_{k \in \mathcal{N}} b^k_{ij} p_k$, and the fact that $z^k_{ij}(A) \leq z^k_{ij}(B)$ implies (A.6). Finally, (A.7) is implied by $a^k_{ij} \leq b^k_{ij}$, which proves the supermodularity of $f_{ij}(A)$. We have

$$
f(A) = \sum_{\{i,j\} \in \mathcal{E}} R_{ij} \Big(f_{ij}(A) + f'_{ij}(A) \Big),
$$

therefore $f(A)$ is supermodular. □

Appendix B

Missing Proofs from Chapter 3

B.1 Proof of Laplacian Update (Lemma 3.4.3)

Let e be the edge chosen for deletion. Observe that deleting e from G results in a rank-one update to L:

$$L' = L - \chi_e c_e \chi_e^T = L - (\sqrt{c_e}\chi_e)(\sqrt{c_e}\chi_e)^T,$$

since $L = BCB^T$. To use Lemma 3.4.2, we need to show that $\sqrt{c_e}\chi_e \in \mathrm{im}(L)$ and $1 - c_e\chi_e^T L^\dagger \chi_e \neq 0$ (as $L \in \mathbb{R}^{n\times n}$ is symmetric). First, since C is a positive definite matrix we can write $L = BCB^T = (BC^{1/2})(BC^{1/2})^T$. Let $U = BC^{1/2}$ and $u = \sqrt{c_e}\chi_e$. By construction u is a column of U, which implies that $u \in \mathrm{im}(U)$. Furthermore, since $\mathrm{im}(U) = \mathrm{im}(UU^T)$, it follows that u is also in $\mathrm{im}(UU^T) = \mathrm{im}(L)$.[1] We further claim that $1 - c_e\chi_e^T L^\dagger \chi_e \neq 0$. To see this, suppose $1 - c_e\chi_e^T L^\dagger \chi_e = 0$. This implies $\chi_e^T L^\dagger \chi_e = 1/c_e$ or $R_{\text{eff}}(e) = r_e$, which happens if and only if e is a bridge, which contradicts the assumption that G' was connected. Now, applying the pseudoinverse update formula given in Lemma 3.4.2 yields the result. □

[1] For any matrix $A \in \mathbb{R}^{m\times n}$, we have $\mathrm{im}(A) = \mathrm{im}(AA^T)$ (see for example Theorem A.25 in [66]).

B.2 Proof of Energy Update (Lemma 3.4.4)

Remember that G_k and G_{k+1} are connected by assumption. Therefore, using Lemma 3.4.3 we have:

$$(L_{k+1})^\dagger = L_k^\dagger + \frac{L_k^\dagger \chi_e c_e \chi_e^T L_k^\dagger}{1 - c_e \chi_e^T L_k^\dagger \chi_e}.$$

Also, the new potentials for G_{k+1} are given by:

$$\begin{aligned}\phi_{k+1} = L_{k+1}^\dagger b = L_k^\dagger b + \frac{L_k^\dagger \chi_e c_e \chi_e^T L_k^\dagger b}{1 - c_e \chi_e^T L_k^\dagger \chi_e} \\ = \phi_k + \frac{L_k^\dagger \chi_e c_e \chi_e^T \phi_k}{1 - c_e \chi_e^T L_k^\dagger \chi_e} = \phi_k + \frac{L_k^\dagger \chi_e f_k(e)}{1 - c_e \chi_e^T L_k^\dagger \chi_e},\end{aligned}$$

where we used the fact that $f_k = C_k B_k^T \phi_k$ (i.e. $f_k(e) = c_e \chi_e^T \phi_k$) in the last equality. By connectivity and feasibility assumptions we know that $b \in \mathrm{im}(L_k)$ and $\chi_e \in \mathrm{im}(L_k)$. This implies that $L_k \phi_k = b$ and similarly, $L_k L_k^\dagger \chi_e = \chi_e$. Then, using these facts and recalling from preliminaries that $\mathcal{E}(f_{k+1}) = b^T \phi_{k+1}$, we have

$$\begin{aligned}\mathcal{E}(f_{k+1}) = b^T \phi_{k+1} = \mathcal{E}(f_k) + \frac{b^T L_k^\dagger \chi_e f_k(e)}{1 - c_e \chi_e^T L_k^\dagger \chi_e} \\ \overset{(a)}{=} \mathcal{E}(f_k) + \frac{(L_k \phi_k)^T L_k^\dagger \chi_e f_k(e)}{1 - c_e R_{\mathrm{eff}}^{(k)}(e)} \overset{(b)}{=} \mathcal{E}(f_k) + \frac{\phi_k^T \chi_e f_k(e)}{1 - c_e R_{\mathrm{eff}}^{(k)}(e)},\end{aligned}$$

where (a) follows from the fact that $L_k \phi_k = b$ and (b) from $L_k L_k^\dagger \chi_e = \chi_e$. The result then follows by noting that from the optimality conditions $\chi_e^T \phi_k = f_k(e)/c_e = r_e f_k(e)$. □

B.3 Stochastic Demands

Here we discuss the extensions of our results for the distribution network reconfiguration problem to the case of stochastic demands. Throughout Chapter 3 we assumed that the demands (d_i) are known and our goal is to find the spanning tree that feeds these demands with the smallest possible loss. However in reality, we may only have access to an estimate of the demand (or its prior distribution) if we are planning ahead for the optimal configuration. Another reason to consider stochastic demands is the actual computation time of finding such optimal configuration. Since the demand changes over time and if the computation time of the optimal spanning tree is not negligible, we may end up with a spanning tree whose loss is not optimal (or approximately optimal) for the current set of demands.

Fortunately, many of the results from Chapter 3 still hold for the case of stochastic demands. This includes the $\mathcal{O}(m-n)$ approximation factor from the RIDE algorithm (Theorem 3.4.1), $\mathcal{O}(n)$ approximation factor of shortest path trees (Theorem 3.5.1), and $\mathcal{O}(\sqrt{n})$ approximation factor for the grids with uniform resistances (Corollary 3.5.3). In fact, none of the previous results requires knowledge of the actual demands in order to find an approximately optimal spanning tree. In RIDE algorithm, we delete edges randomly based on the effective resistances which are independent of the demands. Similarly, shortest path trees are obtained with respect to the edge resistances. Finally in Theorem 3.5.2 and Corollary 3.5.3, after finding a family of cuts that are not too big, the proposed solution is any spanning tree whose edges carry flows

only towards outside of the cuts.

The only algorithm whose performance may be affected by the stochasticity of the demands is the MIN-MIN algorithm. Remember that even in the case of non-uniform demands (Theorem 3.6.6), the MIN-MIN tree would be constructed from the uniform counterpart of the instance. Therefore, we can still run the same algorithm in the case of stochastic demands, and we only need to check how its approximation factor changes based on the accuracy of our demand estimation. Assume that instead of having the actual demand for each vertex (d_i), we have access to an estimate of the demand, denoted by $\hat{d}_i$. Also consider a global parameter $\delta \in (0, 1)$ that measures the accuracy of our estimation. In particular, we assume that the actual demand for each node (d_i) is drawn from a distribution F_i that is supported on $[\hat{d}_i(1-\delta), \hat{d}_i(1+\delta)]$. We have the following result on the performance of the MIN-MIN algorithm.

Theorem B.3.1. *Let $\hat{d}_i$ be the estimate of the demand of node i in a rectangular $n \times m$ $(n \leq m)$ grid with uniform resistances, where the actual demand d_i is drawn from a distribution F_i supported on $[\hat{d}_i(1-\delta), \hat{d}_i(1+\delta)]$. Let $\hat{d}_{\min} = \min_{i \in V \setminus \{r\}} \hat{d}_i$ and $\hat{d}_{\max} = \max_{i \in V \setminus \{r\}} \hat{d}_i$. Then the* MIN-MIN *algorithm gives an approximation factor of $\alpha^2 (\frac{1+\delta}{1-\delta})^2 \left(2 + \mathcal{O}(\frac{1}{\log n})\right)$, where $\alpha = \hat{d}_{\max}/\hat{d}_{\min}$.*

Note that the only difference with Theorem 3.6.6 is the additional $(\frac{1+\delta}{1-\delta})^2$ term. The bigger δ means that our estimates are less accurate, and hence we get a weaker (larger) approximation factor.

Appendix C

Missing Material from Chapter 5

C.1 Bid-oblivious vs Bid-sensitive Mechanisms

In this section, we show an example of how bid-sensitive mechanisms can achieve a lower cost compared to bid-oblivious mechanisms. Remember that our threshold mechanisms determine the number of agents that serve the demand in a dynamic way, after observing the bids. On the other hand, bid-oblivious mechanisms determine this number a-priori, before the bids are submitted.

Example 3 (Bid-oblivious mechanism)**.** Consider 2 rounds and n agents with overhead costs drawn from the uniform distribution, $M_i \sim U[0,1]$. We only have to decide on the number of agents to be saved from round 1 to round 2. If we save 1 agent, the expected cost would be $\frac{2}{n+1}$ for round 1, which is the expected value of the second lowest bid out of n i.i.d. uniform bids in $[0,1]$. However, in that case, there will be no competition in round 2, and we have to pay the upper-bound of the distribution (which is 1) to the single agent left in round 2. On the other hand, if we saved 2 agents in round 1, the expected cost for that round would be $2 \times \frac{3}{n+1}$, because each of the agents gets the third lowest bid. We also have to pay 2/3 in expectation in round 2. Comparing

the two cases ($\frac{2}{n+1} + 1$ versus $2 \times \frac{3}{n+1} + \frac{2}{3}$), we conclude that it is beneficial to save 2 agents if there are more than $n = 11$ agents initially. Similarly, we can compute the number of agents that justify saving 3,4,... agents.

Example 4 (Bid-oblivious vs bid-sensitive)**.** Consider 2 rounds and 2 agents with uniform $[0, 1]$ distribution for overhead costs. According to the previous example, the bid-oblivious mechanism would save only 1 agent in the first round, and therefore pay a total cost of $\frac{2}{3} + 1$ in expectation. However, by setting a threshold of $t_2 = \frac{1}{6}$ (as calculated in Example 1), the expected cost reduces to $\frac{539}{324} \approx 1.663$.

C.2 Dealing with Ties

Allowing for ties in the bids introduces some slight technicalities which we present in the updated definition and proof of truthfulness of our mechanism below.

Definition C.2.1. A single threshold mechanism using thresholds $t_1, \ldots, t_n \in [0, 1]$ is defined as follows: Assume $M_1 \leq M_2 \leq \cdots \leq M_n$ and let us define the predicate $T_k(M_1, \ldots, M_n) = 1$ if and only if $M_k \leq t_k$, in other words the k^{th} smallest value is less than the k^{th} threshold. Let k be the highest index such that $T_k = 1$ and let ℓ be the largest index such that $M_k = M_\ell$. Then the mechanism allocation is:

$$x_i = \begin{cases} 1/\ell & \text{if } i \leq \ell, \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$

and the payment to agent i is $x_i \cdot p$, where p is the total mechanism payment (also the per unit cost of providing the demand) defined as $p = \ell \cdot \min\{t_k, M_{k+1}\}$.

Proposition C.2.1. *Any threshold mechanism is truthful in the corresponding single-shot game and each agent that has non-zero allocation has non-negative utility.*

Proof. If an agent i is not allocated the service, she receives utility of $-\infty$. Bidding a lower overhead cost may result in her being allocated some part of the demand. There are two scenarios in which this may happen: (1) If there exists some k such that T_k is the highest true predicate both before and after agent i lowered her bid. In this case, it must be that her lower bid is less than or equal to M_k, which is strictly less than M_i since if $M_i = M_k$ then agent i should have been allocated. This results in a payment equal to M_k, which makes her utility $-\infty$ again. (2) If T_k is not the highest true predicate after agent i lowers her bid. Assume that the new highest predicate satisfied is T_w for some $w > k$. Since T_w was not true before, it must be that the threshold t_w is now the critical value, therefore each agent receives payment equal to t_w. But since T_w was false before, we know that $t_w < M_i$, meaning that agent i will receive $-\infty$ utility.

If agent i is allocated the service, notice that her payment is independent of her actual overhead cost. Reporting a lower overhead cost does not change her allocation nor payment. Similarly, if she reports a higher amount,

she will receive the same payment, as long as she is still being allocated the service. If her increase makes her not being allocated, then her utility becomes $-\infty$. In neither case is deviating from reporting the true overhead cost profitable. □

C.3 Distributional Assumptions

Throughout the paper, we assumed two important properties for the underlying distribution F: (i) we assumed that the virtual cost is monotone increasing, and (ii) we assumed that the second order statistics have the diminishing returns property, $\mu_{2:n-1} - \mu_{2:n} \geq \mu_{2:n} - \mu_{2:n+1}$. One sufficient condition for (i) is that F is log-concave. For (ii), we can show that: (see for example, [146])

$$\mu_{r:n-1} - \mu_{r:n} = \frac{r}{n}\binom{n}{r}\int_0^1 [F(x)]^r[1-F(x)]^{n-r}dx. \tag{C.2}$$

Therefore, we have to show that the above expression is monotone decreasing in n, for $r = 2$. Since the above integral is hard to compute for arbitrary distributions F, here we show numerically that many distributions satisfy this property. We also compute the above integral for polynomial distributions and show theoretically that they satisfy both of our assumptions.

Figure C.1-(left) shows a truncated normal distribution with $\mu = 0.8$ and $\sigma = 0.1$ (i.e., this is a probability distribution derived from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ by bounding the random variable between 0 and 1). Figure C.1-(middle) shows the marginal decrease in the second order statistic as we increase the number of samples from n to $n+1$, i.e., $\mu_{2:n} - \mu_{n+1}$. As

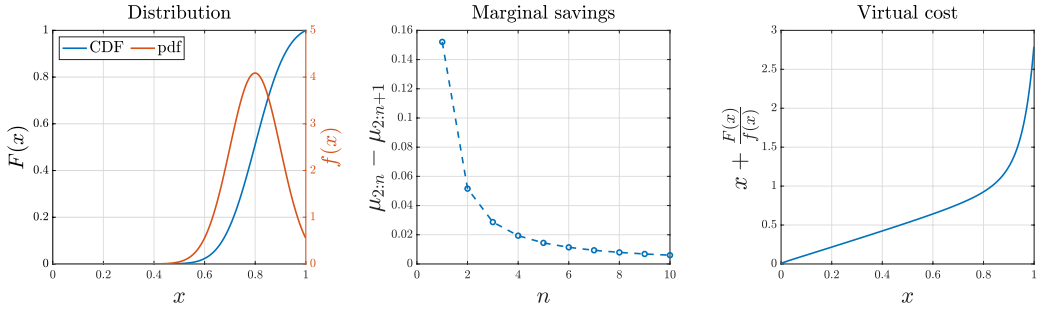

Figure C.1: Regularity of a truncated normal distribution with $\mu = 0.8$ and $\sigma = 0.1$: (Left) Distribution F, (Middle) Marginals of second order statistic, (Right) Virtual cost function.

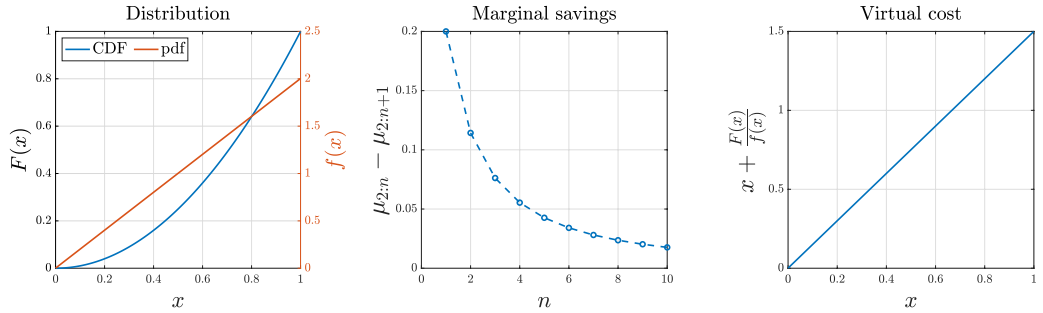

Figure C.2: Regularity of the polynomial distribution with exponent $p = 2$: (Left) Distribution $F(x) = x^2$, (Middle) Marginals of second order statistic, (Right) Virtual cost function.

we can see, the marginal change diminishes as we have more samples. Finally, Figure C.1-(right) shows the virtual cost which confirms its monotonicity.

Figures C.2, C.3 confirm our distributional assumptions for a polynomial distribution $F(x) = x^p$, where $p > 0$. Figure C.2 corresponds to the case of $p = 2$, meaning that the CDF is a quadratic function, $F(x) = x^2$; and Figure C.3 corresponds to $p = 1/2$, meaning that $F(x) = \sqrt{x}$. Note that for polynomial distributions, since $f(x) = px^{p-1}$, we have $\frac{F(x)}{f(x)} = x/p$, which is

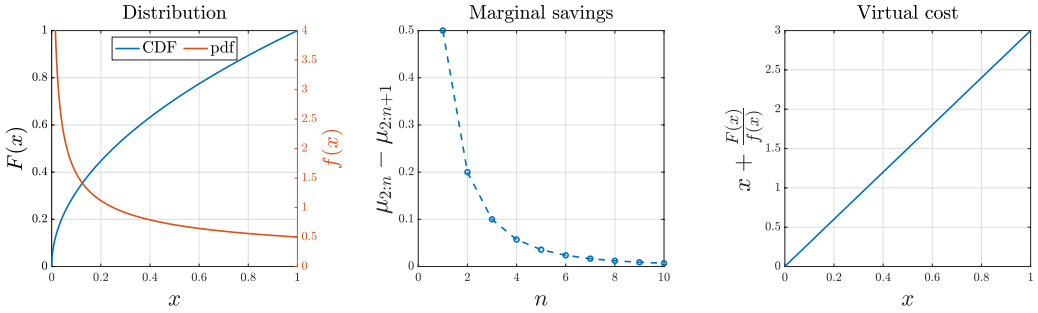

Figure C.3: Regularity of the polynomial distribution with exponent $p = 1/2$: (Left) Distribution $F(x) = \sqrt{x}$, (Middle) Marginals of second order statistic, (Right) Virtual cost function.

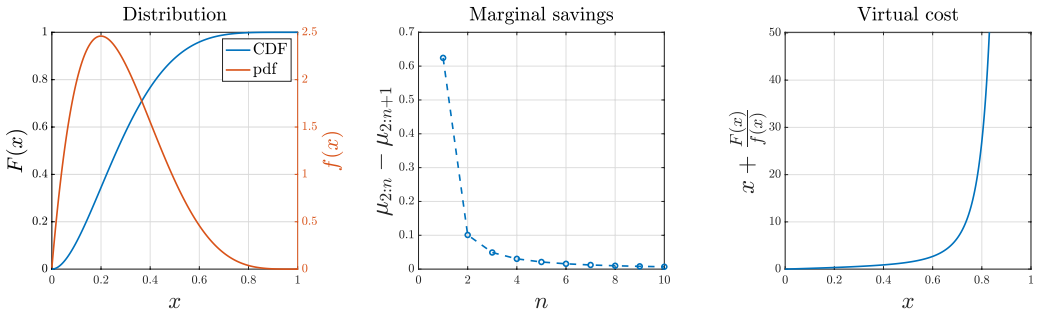

Figure C.4: Regularity of beta distribution with shape parameters $\alpha = 2$ and $\beta = 5$: (Left) Distribution $F(x) = I_x(\alpha, \beta)$, where I denotes the regularized incomplete beta function, (Middle) Marginals of second order statistic, (Right) Virtual cost function.

why the virtual costs are linear in both figures.

Finally, Figure C.4 shows a beta distribution for which we have $f(x) = x^{\alpha-1}(1-x)^{\beta-1}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$, where Γ is the Gamma function. For this figure we have the shape parameters set to $\alpha = 2$ and $\beta = 5$.

We now prove that polynomial distributions satisfy the distributional assumptions needed for our results.

Theorem C.3.1 (polynomial distributions)**.** *For any distribution* $F(x) = x^p, x \in [0,1], p > 0$ *we have:*

(a) $x + \frac{F(x)}{f(x)}$ *is monotone increasing.*

(b) $\mu_{r:n-1} - \mu_{r:n}$ *is monotone decreasing in* n *for any* r*.*

Proof. First, $x + \frac{F(x)}{f(x)} = x + \frac{x^p}{px^{p-1}} = x(1 + \frac{1}{p})$ which is monotone increasing. For the second part, using equation (C.2), we have:

$$\mu_{r:n-1} - \mu_{r:n} = \frac{r}{n}\binom{n}{r}\int_0^1 (x^p)^r(1-x^p)^{n-r}dx = \frac{r}{n}\binom{n}{r}\frac{\Gamma(n+1-r)\Gamma(r+\frac{1}{p})}{p\cdot\Gamma(n+\frac{1}{p}+1)}, \tag{C.3}$$

where Γ is the Gamma function, and satisfies $\Gamma(z+1) = z\Gamma(z)$. Comparing this expression with the same expression for $\mu_{r:n} - \mu_{r:n+1}$, we need to show that:

$$\frac{(n-1)!}{(r-1)!(n-r)!} \times \frac{\Gamma(n+1-r)\Gamma(r+\frac{1}{p})}{p\cdot\Gamma(n+\frac{1}{p}+1)} \geq \frac{n!}{(r-1)!(n+1-r)!} \times \frac{\Gamma(n+2-r)\Gamma(r+\frac{1}{p})}{p\cdot\Gamma(n+\frac{1}{p}+2)},$$

or equivalently

$$\frac{\Gamma(n+\frac{1}{p}+2)}{\Gamma(n+\frac{1}{p}+1)} \geq \frac{n}{n+1-r} \times \frac{\Gamma(n+2-r)}{\Gamma(n+1-r)}.$$

Now using the above-mentioned property of the Gamma function $\Gamma(z+1) = z\Gamma(z)$, notice that the left hand side is equal to $n+1+1/p$, while the right hand side is equal to n; therefore, the inequality is true for any $p > 0$. □

C.4 Non-monotone Example

Here we give an example where the global optimal thresholds are non-monotone, in particular for this example $t_2^* < t_3^*$. We proved via Lemma 5.5.1 and Theorem 5.5.5 that this cannot happen if the underlying distribution F satisfies our two assumptions of regularity (Definition 5.3.3) and diminishing returns of order statistics (Definition 5.3.5). In fact, the distribution F in the following example fails both of these assumptions. It has a point mass probability which breaks the monotonicity of the virtual cost, and more importantly, its second order statistic does not have the diminishing returns property. More precisely, the savings of having the 3rd agent participating in the second day ($\mu_{2:2} - \mu_{2:3}$) is more than the savings from the 2nd agent ($1 - \mu_{2:2}$). This makes the buyer want to save the 3rd agent even at a higher value than he is willing to pay for the 2nd agent.

Example 5 (Non-monotone thresholds)**.** Consider a distribution F that is $U[0, \frac{1}{2}]$ with probability 1/2 and equal to 1 with probability 1/2. In other words,

$$M = \begin{cases} X & \text{w.p. } \frac{1}{2}, \\ 1 & \text{w.p. } \frac{1}{2}, \end{cases}$$

where $X \sim U[0, \frac{1}{2}]$. Now suppose we have three agents with overhead costs drawn independently from F. If we save 1, 2, or 3 agents, the expected cost we incur on day 2 would be 1, $\mu_{2:2} = \frac{5}{6}$, and $\mu_{2:3} = \frac{21}{32}$, respectively. Additionally, the optimal thresholds on day 1 are

$$t_2^* = \frac{1}{2}(1 - \mu_{2:2}) = \frac{1}{12} \approx 0.083,$$

$$t_3^* = \frac{1}{2}(\mu_{2:2} - \mu_{2:3}) = \frac{17}{192} \approx 0.088,$$

which show that the optimal thresholds can be non-monotone ($t_3^* > t_2^*$), if we do not have constraints on the underlying distribution F.

C.5 Proof of Theorem 5.4.1

Using linearity of expectation and law of total probability we have:

$$\begin{aligned}
&\frac{\partial C_n(t_2, ..., t_n)}{\partial t_i} = \frac{\partial}{\partial t_i}\mathbb{E}_{\mathbf{M}}\left[g(\mathbf{M}, t_2, ..., t_n)\right] = \mathbb{E}_{\mathbf{M}}\left[\frac{\partial g(\mathbf{M}, t_2, ..., t_n)}{\partial t_i}\right] \\
&= \sum_{j \geq i}^{n} \Pr(T_j = 1, T_{j+1} = ... = T_n = 0)\mathbb{E}_{\mathbf{M}}\left[\frac{\partial g(\mathbf{M}, t_2, ..., t_n)}{\partial t_i} \mid T_j = 1, T_{j+1} = ... = T_n = 0\right] \\
&+ \Pr(T_i = ... = T_n = 0)\mathbb{E}_{\mathbf{M}}\left[\frac{\partial g(\mathbf{M}, t_2, ..., t_n)}{\partial t_i} \mid T_i = ... = T_n = 0\right] \qquad \text{(C.4)}
\end{aligned}$$

Note that for $j \geq i + 1$, the derivative (and therefore the conditional expectation) is zero because we save at least $i + 1$ agents and the payment is independent of t_i. For $j = i$, if the $(i + 1)^{\text{th}}$ bid (i.e., $M_{i+1:n}$) is more than t_i, then we have $g(\mathbf{M}, t_2, ..., t_n) = i \times t_i + \mu_{2:i}$, and hence

$$\mathbb{E}_{\mathbf{M}}\left[\frac{\partial g(\mathbf{M}, t_2, ..., t_n)}{\partial t_i} \mid T_i = 1, T_{i+1} = ... = T_n = 0, M_{i+1:n} > t_i\right] = i.$$

In addition, the corresponding probability of such an event is

$$\begin{aligned}
&\Pr(T_i = 1, T_{i+1} = ... = T_n = 0, M_{i+1:n} > t_i) \\
&= \binom{n}{i} F(t_i)^i \Pr\left[M_{i+1}, ..., M_n > t_i, T_{i+1} = ... = T_n = 0 \mid M_1, ..., M_i \leq t_i\right] \\
&= \binom{n}{i} F(t_i)^i P_{i,n} \qquad \text{(C.5)}
\end{aligned}$$

For the case of $T_i = ... = T_n = 0$, we save at most $i-1$ agents and the derivative of the cost with respect to t_i is zero, unless there is a bid in $(t_i, t_i + \epsilon)$ such that increasing t_i by ϵ makes $T_i = 1$. In this special case we use the fact that:

$$\frac{\partial g(\mathbf{M}, t_2, ..., t_n)}{\partial t_i} = \lim_{\epsilon \to 0} \frac{g(\mathbf{M}, t_2, ..., t_i + \epsilon, ..., t_n) - g(\mathbf{M}, t_2, ..., t_i, ..., t_n)}{\epsilon}.$$

Note that this derivative would be infinity as we have a sudden increase in g, however the probability of this event is proportional to ϵ which makes a finite product. In particular, we have:

$$\begin{aligned} &\Pr(T_i = ... = T_n = 0, t_i < M_{i:n} < t_i + \epsilon) \qquad \text{(C.6)} \\ &= n\binom{n-1}{i-1} F(t_i)^{i-1} \epsilon f(t_i) \\ &\qquad \times \Pr\left[M_{i+1}, ..., M_n > t_i + \epsilon, T_{i+1} = ... = T_n = 0 \mid M_1, ..., M_i \leq t_i + \epsilon\right] \\ &= n\binom{n-1}{i-1} F(t_i)^{i-1} \epsilon f(t_i) P_{i,n} \end{aligned}$$

In this case, $g(\mathbf{M}, t_2, ..., t_i + \epsilon, ..., t_n) = i \times (t_i + \epsilon) + \mu_{2:i}$. Even though the cost $g(\mathbf{M}, t_2, ..., t_i, ..., t_n)$ depends on the thresholds $t_2, ..., t_i$ and the realization of the bids, its expectation is independent of the number of agents n, since we have already conditioned on the fact that exactly $i-1$ bids are below t_i (and the assignment and payments only depend on those $i-1$ bids). Putting the results back into the original summation (C.4), we get:

$$\begin{aligned} &\frac{\partial C_n(t_2, ..., t_n)}{\partial t_i} = i \times \binom{n}{i} F(t_i)^i P_{i,n} + n\binom{n-1}{i-1} F(t_i)^{i-1} f(t_i) P_{i,n} \times \qquad \text{(C.7)} \\ &\mathbb{E}_{\mathbf{M}}\left[i \times t_i + \mu_{2:i} - g(\mathbf{M}, t_2, ..., t_n) \mid T_i = ... = T_n = 0, t_i < M_{i:n} < t_i + \epsilon\right] \end{aligned}$$

Using the identity $i\binom{n}{i} = n\binom{n-1}{i-1}$ we get:

$$\frac{\partial C_n(t_2, ..., t_n)}{\partial t_i} = i\binom{n}{i} P_{i,n} F(t_i)^{i-1} f(t_i) \times \tag{C.8}$$
$$\left[\frac{F(t_i)}{f(t_i)} + \mathbb{E}_{\mathbf{M}}\big[i \times t_i + \mu_{2:i} - g(\mathbf{M}, t_2, ..., t_n) \mid T_i = ... = T_n = 0, t_i < M_{i:n} < t_i + \epsilon\big]\right],$$

which completes the proof. □

Bibliography

[1] S. Civanlar, J. Grainger, H. Yin, and S. Lee, "Distribution feeder reconfiguration for loss reduction," *IEEE Transactions on Power Delivery*, vol. 3, no. 3, pp. 1217–1223, 1988.

[2] M. E. Baran and F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1401–1407, 1989.

[3] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.

[4] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2002, vol. 24.

[5] A. Khodabakhsh, G. Yang, S. Basu, E. Nikolova, M. C. Caramanis, T. Lianeas, and E. Pountourakis, "A submodular approach for electricity distribution network reconfiguration," in *51st Hawaii International Conference on System Sciences (HICSS)*, 2018, pp. 2717–2726.

[6] K. Sathish Kumar and S. Naveen, "Power system reconfiguration and loss minimization for a distribution system using catfish PSO algorithm," *Frontiers in Energy*, vol. 8, no. 4, pp. 434–442, 2013.

[7] R. J. Sarfi, M. Salama, and A. Chikhani, "A survey of the state of the art in distribution system reconfiguration for system loss reduction," *Electric Power Systems Research*, vol. 31, no. 1, pp. 61–70, 1994.

[8] E. A. Goldis, X. Li, M. C. Caramanis, A. M. Rudkevich, and P. A. Ruiz, "AC-Based topology control algorithms (TCA)–A PJM historical data case study," in *IEEE 48th Hawaii International Conference on System Sciences (HICSS)*, 2015, pp. 2516–2519.

[9] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman and Company, New York, 1979.

[10] Z. Liu, A. Clark, P. Lee, L. Bushnell, D. Kirschen, and R. Poovendran, "Towards scalable voltage control in smart grid: a submodular optimization approach," in *Proceedings of the 7th International Conference on Cyber-Physical Systems*, 2016, p. 20.

[11] M. G. Damavandi, V. Krishnamurthy, and J. R. Martí, "Robust meter placement for state estimation in active distribution systems," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1972–1982, 2015.

[12] N. Gensollen, V. Gauthier, M. Marot, and M. Becker, "Submodular optimization for control of prosumer networks," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2016, pp. 180–185.

[13] H. Khodr and J. Martinez-Crespo, "Integral methodology for distribution systems reconfiguration based on optimal power flow using benders decomposition technique," *IET Generation, Transmission & Distribution*, vol. 3, no. 6, pp. 521–534, 2009.

[14] E. Míguez, J. Cidrás, E. Díaz-Dorado, and J. L. García-Dornelas, "An improved branch-exchange algorithm for large-scale distribution network planning," *IEEE Transactions on Power Systems*, vol. 17, no. 4, pp. 931–936, 2002.

[15] Q. Peng and S. H. Low, "Optimal branch exchange for feeder reconfiguration in distribution networks," in *IEEE 52nd Annual Conference on Decision and Control (CDC)*, 2013, pp. 2960–2965.

[16] F. V. Gomes, S. Carneiro, J. L. R. Pereira, M. P. Vinagre, P. A. N. Garcia, and L. R. Araujo, "A new heuristic reconfiguration algorithm for large distribution systems," *IEEE Transactions on Power systems*, vol. 20, no. 3, pp. 1373–1378, 2005.

[17] D. Shirmohammadi and H. W. Hong, "Reconfiguration of electric distribution networks for resistive line losses reduction," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1492–1498, 1989.

[18] T. E. McDermott, I. Drezga, and R. P. Broadwater, "A heuristic nonlinear constructive method for distribution system reconfiguration," *IEEE Transactions on Power Systems*, vol. 14, no. 2, pp. 478–483, 1999.

[19] A. B. Morton and I. M. Mareels, "An efficient brute-force solution to the network reconfiguration problem," *IEEE Transactions on Power Delivery*, vol. 15, no. 3, pp. 996–1000, 2000.

[20] W. Kocay and D. L. Kreher, *Graphs, Algorithms, and Optimization*. Chapman and Hall/CRC, 2016.

[21] W. Lin, F. Cheng, and M. Tsay, "Distribution feeder reconfiguration with refined genetic algorithm," *IEEE Proceedings-Generation, Transmission and Distribution*, vol. 147, no. 6, pp. 349–354, 2000.

[22] B. Enacheanu, B. Raison, R. Caire, O. Devaux, W. Bienia, and N. HadjSaid, "Radial network reconfiguration using genetic algorithm based on the matroid theory," *IEEE Transactions on Power Systems*, vol. 23, no. 1, pp. 186–195, 2008.

[23] Y. K. Wu, C. Y. Lee, L. C. Liu, and S. H. Tsai, "Study of reconfiguration for the distribution system with distributed generators," *IEEE transactions on Power Delivery*, vol. 25, no. 3, pp. 1678–1685, 2010.

[24] H. Kim, Y. Ko, and K. Jung, "Artificial neural-network based feeder reconfiguration for loss reduction in distribution systems," *IEEE Transactions on Power Delivery*, vol. 8, no. 3, pp. 1356–1366, 1993.

[25] M. Kashem, G. Jasmon, A. Mohamed, and M. Moghavvemi, "Artificial neural network approach to network reconfiguration for loss minimiza-

tion in distribution networks," *International Journal of Electrical Power & Energy Systems*, vol. 20, no. 4, pp. 247–258, 1998.

[26] M. E. Baran and F. F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Transactions on power Delivery*, vol. 4, no. 1, pp. 725–734, 1989.

[27] R. K. Martin, "A sharp polynomial size linear programming formulation of the minimum spanning tree problem," *Graduate School of Business, University of Chicago, Chicago, IL*, 1986.

[28] S. Raghavan, "Formulations and algorithms for network design problems with connectivity requirements," Ph.D. dissertation, Massachusetts Institute of Technology, 1994.

[29] S. Mittal and A. S. Schulz, "An FPTAS for optimizing a class of low-rank functions over a polytope," *Mathematical Programming*, pp. 1–18, 2013.

[30] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proceedings of the 41st Annual ACM symposium on Theory of Computing*. ACM, 2009, pp. 323–332.

[31] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power

systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2011.

[32] D. Shirmohammadi and H. W. Hong, "Reconfiguration of electric distribution networks for resistive line losses reduction," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1492–1498, 1989.

[33] S. Gupta, A. Khodabakhsh, H. Mortagy, and E. Nikolova, "Electrical flows over spanning trees," *Mathematical Programming*, pp. 1–41, 2021.

[34] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The QC relaxation: A theoretical and computational study on optimal power flow," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 3008–3018, 2015.

[35] L. Gan, N. Li, U. Topcu, and S. H. Low, "Exact convex relaxation of optimal power flow in radial networks," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 72–87, 2014.

[36] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Foundations and Trends in Electric Energy Systems*, vol. 4, no. 1-2, pp. 1–221, 2019.

[37] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, "Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, 2011, pp. 273–282.

[38] S. H. Low, "Convex relaxation of optimal power flow—part I: Formulations and equivalence," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 15–27, 2014.

[39] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: Mesh networks," *IEEE Transactions on Power Systems*, vol. 30, no. 1, pp. 199–211, 2014.

[40] D. K. Molzahn, B. C. Lesieutre, and C. L. DeMarco, "Investigation of non-zero duality gap solutions to a semidefinite relaxation of the optimal power flow problem," in *Proceedings of the 47th Hawaii International Conference on System Sciences*, 2014, pp. 2325–2334.

[41] Q. Peng and S. H. Low, "Distributed optimal power flow algorithm for radial networks, I: Balanced single phase case," *IEEE Transactions on Smart Grid*, vol. 9, no. 1, pp. 111–121, 2016.

[42] M. Baes, A. Del Pia, Y. Nesterov, S. Onn, and R. Weismantel, "Minimizing Lipschitz-continuous strongly convex functions over integer points in polytopes," *Mathematical Programming*, vol. 134, no. 1, p. 305?322, 2012.

[43] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*, 1st ed. Athena Scientific, 1997.

[44] J. Haddock, "Projection algorithms for convex and combinatorial optimization," Ph.D. dissertation, University of California, Davis, 2018.

[45] E. B. Fisher, R. P. O'Neill, and M. C. Ferris, "Optimal transmission switching," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1346–1355, 2008.

[46] S. Fattahi, J. Lavaei, and A. Atamtürk, "A bound strengthening method for optimal transmission switching in power systems," *IEEE Transactions on Power Systems*, vol. 34, no. 1, pp. 280–291, 2018.

[47] B. Kocuk, S. S. Dey, and X. A. Sun, "New formulation and strong MISOCP relaxations for AC optimal transmission switching problem," *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4161–4170, 2017.

[48] A. Grastien, I. Rutter, D. Wagner, F. Wegner, and M. Wolf, "The maximum transmission switching flow problem," in *Proceedings of the 9th International Conference on Future Energy Systems*. ACM, 2018, pp. 340–360.

[49] B. Kocuk, H. Jeon, S. S. Dey, J. Linderoth, J. Luedtke, and X. A. Sun, "A cycle-based formulation and valid inequalities for DC power transmission problems with switching," *Operations Research*, vol. 64, no. 4, pp. 922–938, 2016.

[50] K. Lehmann, A. Grastien, and P. Van Hentenryck, "The complexity of DC-switching problems," *arXiv preprint arXiv:1411.4369*, 2014.

[51] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu, "Solving SDD linear systems in nearly $m \log^{1/2} n$

time," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, 2014, pp. 343–352.

[52] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A simple, combinatorial algorithm for solving SDD systems in nearly-linear time," in *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, 2013, pp. 911–920.

[53] M. B. Cohen, J. A. Kelner, R. Kyng, J. Peebles, R. Peng, A. B. Rao, and A. Sidford, "Solving directed Laplacian systems in nearly-linear time through sparse LU factorizations," in *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science*, 2018, pp. 898–909.

[54] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011.

[55] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[56] M. Elkin, Y. Emek, D. A. Spielman, and S.-H. Teng, "Lower-stretch spanning trees," in *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, 2005, pp. 494–503.

[57] I. Abraham and O. Neiman, "Using petal-decompositions to build a low

stretch spanning tree," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, 2012, pp. 395–406.

[58] I. Abraham, Y. Bartal, and O. Neiman, "Nearly tight low stretch spanning trees," in *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008, pp. 781–790.

[59] N. Alon, R. M. Karp, D. Peleg, and D. West, "A graph-theoretic game and its application to the k-server problem," *SIAM Journal on Computing*, vol. 24, no. 1, pp. 78–100, 1995.

[60] A. Madry, "Computing maximum flow with augmenting electrical flows," in *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science*, 2016, pp. 593–602.

[61] N. Anari and S. Oveis Gharan, "Effective-resistance-reducing flows, spectrally thin trees, and asymmetric TSP," in *Proceedings of the 56th IEEE Annual Symposium on Foundations of Computer Science*, October 2015, pp. 20–39.

[62] D. P. Williamson, *Network Flow Algorithms*. Cambridge University Press, 2019.

[63] R. Lyons and Y. Peres, *Probability on Trees and Networks*. Cambridge University Press, 2017.

[64] D. Durfee, R. Kyng, J. Peebles, A. B. Rao, and S. Sachdeva, "Sampling random spanning trees faster than matrix multiplication," in *Proceedings*

of the 49th Annual ACM Symposium on Theory of Computing, 2017, pp. 730–742.

[65] D. A. Levin and Y. Peres, *Markov Chains and Mixing Times*. American Mathematical Society., 2017, vol. 107.

[66] C. R. Rao and H. Toutenburg, *Linear Models Least Squares and Alternatives*, 2nd ed. Springer, 1999.

[67] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: a network design problem for multicommodity flow," in *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*. ACM, 2001, pp. 389–398.

[68] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Springer Science & Business Media, 2003, vol. 24.

[69] R. J. Lipton and R. E. Tarjan, "A separator theorem for planar graphs," *SIAM Journal on Applied Mathematics*, vol. 36, no. 2, pp. 177–189, 1979.

[70] R. Martin, "Using separation algorithms to generate mixed integer model reformulations," *Operations Research Letters*, vol. 10, no. 3, pp. 119–128, 1991.

[71] L. Gurobi Optimization, "Gurobi optimizer reference manual version 9.0," 2020, uRL: https://www.gurobi.com/documentation/9.0/refman.

[72] A. Khodabakhsh, J. Matuschke, J. Horn, E. Nikolova, and E. Pountourakis, "Electric vehicle valet," *arXiv preprint arXiv:1811.06184*, 2018.

[73] C. Kennedy. (2020) The most dramatic year in the history of oil. [Online]. Available: https://oilprice.com/Energy/Oil-Prices/The-Most-Dramatic-Year-In-The-History-Of-Oil.html

[74] T. Paraskova. (2020) Tesla could launch a million-mile battery this year. [Online]. Available: https://oilprice.com/Energy/Energy-General/Tesla-Could-Launch-A-Million-Mile-Battery-This-Year.html

[75] ——. (2020) Tesla's newest rival is taking the stock market by storm. [Online]. Available: https://oilprice.com/Energy/Energy-General/Teslas-Newest-Rival-Is-Taking-The-Stock-Market-By-Storm.html

[76] K. Korosec. (2019) Tesla plans to launch a robotaxi network in 2020. [Online]. Available: https://techcrunch.com/2019/04/22/tesla-plans-to-launch-a-robotaxi-network-in-2020/

[77] M. Nick, R. Cherkaoui, and M. Paolone, "Optimal allocation of dispersed energy storage systems in active distribution networks for energy balance and grid support," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2300–2310, 2014.

[78] Y. Wang, K. Tan, X. Y. Peng, and P. L. So, "Coordinated control of distributed energy-storage systems for voltage regulation in distribution

networks," *IEEE transactions on power delivery*, vol. 31, no. 3, pp. 1132–1141, 2015.

[79] J. Teng, S. Luan, D. Lee, and Y. Huang, "Optimal charging/discharging scheduling of battery storage systems for distribution systems interconnected with sizeable PV generation systems," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1425–1433, 2012.

[80] S. Wen, H. Lan, Q. Fu, C. Y. David, and L. Zhang, "Economic allocation for energy storage system considering wind power distribution," *IEEE Transactions on power Systems*, vol. 30, no. 2, pp. 644–652, 2014.

[81] R. Yu, W. Zhong, S. Xie, C. Yuen, S. Gjessing, and Y. Zhang, "Balancing power demand through EV mobility in vehicle-to-grid mobile energy networks," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 79–90, 2015.

[82] R. Johnson, M. Mayfield, and S. Beck, "Utilization of stochastically located customer owned battery energy storage systems for violation management on UK LV residential feeders with varying renewables penetrations," *Journal of Energy Storage*, vol. 19, pp. 52–66, 2018.

[83] ——, "Optimal placement, sizing, and dispatch of multiple BES systems on UK low voltage residential networks," *Journal of Energy Storage*, vol. 17, pp. 272–286, 2018.

[84] ——, "Battery energy storage for management of LV network operational violations: a multi-feeder analysis," *Energy Procedia*, vol. 151, pp. 31–36, 2018.

[85] M. Singh, I. Kar, and P. Kumar, "Influence of EV on grid power quality and optimizing the charging schedule to mitigate voltage imbalance and reduce power loss," in *IEEE 14th International Power Electronics and Motion Control Conference (EPE/PEMC)*, 2010, pp. T2–196.

[86] K. Clement-Nyns, E. Haesen, and J. Driesen, "The impact of charging plug-in hybrid electric vehicles on a residential distribution grid," *IEEE Transactions on power systems*, vol. 25, no. 1, pp. 371–380, 2010.

[87] Q. Xiang, F. Kong, X. Liu, X. Chen, L. Kong, and L. Rao, "Auc2charge: An online auction framework for electric vehicle park-and-charge," in *Proceedings of the Sixth ACM International Conference on Future Energy Systems*. ACM, 2015, pp. 151–160.

[88] H. Liu, Z. Hu, Y. Song, and J. Lin, "Decentralized vehicle-to-grid control for primary frequency regulation considering charging demands," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 3480–3489, 2013.

[89] J. Y. Yong, V. K. Ramachandaramurthy, K. M. Tan, and N. Mithulananthan, "Bi-directional electric vehicle fast charging station with novel reactive power compensation for voltage regulation," *International Journal of Electrical Power & Energy Systems*, vol. 64, pp. 300–310, 2015.

[90] W. Kempton and S. E. Letendre, "Electric vehicles as a new power source for electric utilities," *Transportation research. Part D, Transport and environment*, vol. 2, no. 3, pp. 157–175, 1997.

[91] M. A. Hussain, W. Brandauer, and M. J. Lee, "Mobility-aware vehicle-to-grid (V2G) optimization for uniform utilization in smart grid based power distribution network," *Mobile Networks and Applications*, pp. 1–12, 2018.

[92] G. Zhang, T. Tan, and G. Wang, "Real-time smart charging of electric vehicles for demand charge reduction at non-residential sites," *IEEE Transactions on Smart Grid*, 2017.

[93] L. Gan, U. Topcu, and S. H. Low, "Optimal decentralized protocol for electric vehicle charging," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 940–951, 2013.

[94] M. Caramanis and J. M. Foster, "Management of electric vehicle charging to mitigate renewable generation intermittency and distribution network congestion," in *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009, pp. 4717–4722.

[95] M. Amjad, A. Ahmad, M. H. Rehmani, and T. Umer, "A review of EVs charging: From the perspective of energy optimization, optimization approaches, and charging techniques," *Transportation Research Part D: Transport and Environment*, vol. 62, pp. 386–417, 2018.

[96] Y. He, B. Venkatesh, and L. Guan, "Optimal scheduling for charging and discharging of electric vehicles," *IEEE transactions on smart grid*, vol. 3, no. 3, pp. 1095–1105, 2012.

[97] C. Hutson, G. K. Venayagamoorthy, and K. A. Corzine, "Intelligent scheduling of hybrid and electric vehicle storage capacity in a parking lot for profit maximization in grid power transactions," in *IEEE Energy 2030 Conference*, 2008, pp. 1–8.

[98] J. Timpner and L. Wolf, "Design and evaluation of charging station scheduling strategies for electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 579–588, 2014.

[99] H. H. Abdeltawab and Y. A. Mohamed, "Mobile energy storage scheduling and operation in active distribution systems," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 9, pp. 6828–6840, 2017.

[100] A. Krishna, A. Narayanan, S. Krishnakumar, P. Misra, A. Vasan, V. Sarangan, and A. Sivasubramaniam, "Uberizing the charging ecosystem for electric vehicles," in *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, 2020, pp. 156–160.

[101] B. Sun, T. Li, S. H. Low, and D. H. Tsang, "ORC: An online competitive algorithm for recommendation and charging schedule in electric vehicle charging network," in *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, 2020, pp. 144–155.

[102] S. Lei, C. Chen, Y. Li, and Y. Hou, "Resilient disaster recovery logistics of distribution systems: Co-optimize service restoration with repair crew and mobile power source dispatch," *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6187–6202, 2019.

[103] B. Zhou, D. Xu, C. Li, Y. Cao, K. W. Chan, Y. Xu, and M. Cao, "Multiobjective generation portfolio of hybrid energy generating station for mobile emergency power supplies," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 5786–5797, 2017.

[104] S. Yao, P. Wang, and T. Zhao, "Transportable energy storage for more resilient distribution systems with multiple microgrids," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 3331–3341, 2018.

[105] C. Jin, J. Tang, and P. Ghosh, "Optimizing electric vehicle charging with energy storage in the electricity market," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 311–320, 2013.

[106] M. G. Vayá and G. Andersson, "Centralized and decentralized approaches to smart charging of plug-in vehicles," in *IEEE Power and Energy Society General Meeting*, 2012, pp. 1–8.

[107] A. Y. Saber and G. K. Venayagamoorthy, "Optimization of vehicle-to-grid scheduling in constrained parking lots," in *IEEE Power & Energy Society General Meeting*, 2009, pp. 1–8.

[108] J. Soares, T. Sousa, H. Morais, Z. Vale, and P. Faria, "An optimal scheduling problem in distribution networks considering V2G," in *IEEE Symposium on Computational Intelligence Applications In Smart Grid (CIASG)*, 2011, pp. 1–8.

[109] S. Yang, M. Wu, X. Yao, and J. Jiang, "Load modeling and identification based on ant colony algorithms for EV charging stations," *IEEE Transactions on Power Systems*, vol. 30, no. 4, pp. 1997–2003, 2015.

[110] P. Van Hentenryck and C. Coffrin, "Transmission system repair and restoration," *Mathematical Programming*, vol. 151, no. 1, pp. 347–373, 2015.

[111] Y. Tan, F. Qiu, A. K. Das, D. S. Kirschen, P. Arabshahi, and J. Wang, "Scheduling post-disaster repairs in electricity distribution networks," *IEEE Transactions on Power Systems*, vol. 34, no. 4, pp. 2611–2621, 2019.

[112] J. Zhai, S. C. Chau, and M. Chen, "Stay or switch: Competitive online algorithms for energy plan selection in energy markets with retail choice," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, 2019, pp. 100–110.

[113] Y. Chen, S. Mei, W. Wei, S. H. Low, A. Wierman, and F. Liu, "Buy or sell? energy sharing of prosumers on constrained networks," *arXiv preprint arXiv:1906.09891*, 2019.

[114] A. Khodabakhsh, J. Horn, E. Nikolova, and E. Pountourakis, "Prosumer pricing, incentives and fairness," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, 2019, pp. 116–120.

[115] Q. Liu, H. Zeng, and M. Chen, "Energy-efficient timely truck transportation for geographically-dispersed tasks," in *Proceedings of the Ninth International Conference on Future Energy Systems*, 2018, pp. 324–339.

[116] Z. Liu, A. Clark, P. Lee, L. Bushnell, D. Kirschen, and R. Poovendran, "Towards scalable voltage control in smart grid: A submodular optimization approach," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–10.

[117] H. N. Ballouz, Private Communication, 13001 W. Highway 71, Austin, TX 78738, 2018. [Online]. Available: http://www.epeconsulting.com

[118] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. WH Freeman and Company, San Francisco, 1979.

[119] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[120] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[121] R. Bhatia, J. Chuzhoy, A. Freund, and J. S. Naor, "Algorithmic aspects of bandwidth trading," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 1, p. 10, 2007.

[122] A. Khodabakhsh, E. Nikolova, E. Pountourakis, and J. Horn, "Dynamic procurement auction with abandonment," *Unpublished manuscript*, 2021.

[123] V. Krishna, *Auction theory*. Academic press, 2009.

[124] E. Bosio and S. Djankov, "How large is public procurement?" 2020, https://blogs.worldbank.org/developmenttalk/how-large-public-procurement, accessed June 5, 2020. [Online]. Available: https://blogs.worldbank.org/developmenttalk/how-large-public-procurement

[125] S. Agrawal, C. Daskalakis, V. Mirrokni, and B. Sivan, "Robust repeated auctions under heterogeneous buyer behavior," in *19th ACM conference on Economics and Computation*, 2018.

[126] V. Saini, "Reserve prices in a dynamic auction when bidders are capacity-constrained," *Economics Letters*, vol. 108, no. 3, pp. 303–306, 2010.

[127] ——, "Endogenous asymmetry in a dynamic procurement auction," *The RAND Journal of Economics*, vol. 43, no. 4, pp. 726–760, 2012.

[128] J. Farrell and P. Klemperer, "Coordination and lock-in: Competition with switching costs and network effects," *Handbook of industrial organization*, vol. 3, pp. 1967–2072, 2007.

[129] T. R. Lewis and H. Yildirim, "Managing switching costs in multiperiod procurements with strategic buyers," *International Economic Review*, vol. 46, no. 4, pp. 1233–1269, 2005.

[130] S. Liu, C. Liu, and Q. Hu, "Optimal procurement strategies by reverse auctions with stochastic demand," *Economic Modelling*, vol. 35, pp. 430–435, 2013.

[131] G. Van Ryzin and G. Vulcano, "Optimal auctioning and ordering in an infinite horizon inventory-pricing system," *Operations Research*, vol. 52, no. 3, pp. 346–367, 2004.

[132] T. R. Lewis and H. Yildirim, "Managing dynamic competition," *American Economic Review*, vol. 92, no. 4, pp. 779–797, 2002.

[133] A. Pavan, I. Segal, and J. Toikka, "Dynamic mechanism design: A myersonian approach," *Econometrica*, vol. 82, no. 2, pp. 601–653, 2014.

[134] J. J. Anton and D. A. Yao, "Second sourcing and the experience curve: price competition in defense procurement," *The RAND Journal of Economics*, pp. 57–76, 1987.

[135] H. Yildirim, "Piecewise procurement of a large-scale project," *International Journal of Industrial Organization*, vol. 22, no. 8-9, pp. 1349–1375, 2004.

[136] J. Horn, Y. Wu, A. Khodabakhsh, E. Nikolova, and E. Pountourakis, "The long-term cost of energy generation," in *Proceedings of the Eleventh*

ACM International Conference on Future Energy Systems, 2020, pp. 74–85.

[137] D. F. Garrett, "Intertemporal price discrimination: Dynamic arrivals and changing values," *American Economic Review*, vol. 106, no. 11, pp. 3275–99, November 2016. [Online]. Available: https://www.aeaweb.org/articles?id=10.1257/aer.20130564

[138] ——, "Dynamic mechanism design: Dynamic arrivals and changing values," *Games and Economic Behavior*, vol. 104, pp. 595 – 612, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0899825617300696

[139] S. Chawla, D. L. Malec, and A. Malekian, "Bayesian mechanism design for budget-constrained agents," in *Proceedings of the 12th ACM conference on Electronic commerce*, 2011, pp. 253–262.

[140] J. Feldman, S. Muthukrishnan, E. Nikolova, and M. Pál, "A truthful mechanism for offline ad slot scheduling," in *International Symposium on Algorithmic Game Theory*. Springer, 2008, pp. 182–193.

[141] J. D. Hartline, "Mechanism design and approximation," 2011.

[142] I. Ashlagi, C. Daskalakis, and N. Haghpanah, "Sequential mechanisms with ex-post participation guarantees," in *Proceedings of the 2016 ACM Conference on Economics and Computation*, 2016, pp. 213–214.

[143] C. H. Papadimitriou, G. Pierrakos, C. Psomas, and A. Rubinstein, "On the complexity of dynamic mechanism design," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, 2016, pp. 1458–1475. [Online]. Available: https://doi.org/10.1137/1.9781611974331.ch100

[144] D. Bergemann and J. Välimäki, "Dynamic mechanism design: An introduction," *Journal of Economic Literature*, vol. 57, no. 2, pp. 235–74, 2019.

[145] F. Chen, "Auctioning supply contracts," *Management Science*, vol. 53, no. 10, pp. 1562–1576, 2007.

[146] H. A. David and H. N. Nagaraja, *Order statistics*. John Wiley & Sons, 2004.

Vita

Ali Khodabakhsh earned his Bachelor's degree in Electrical Engineering (Communication Systems) from Sharif University of Technology in 2014, and the M.Sc. degree in Electrical and Computer Engineering from the University of Texas at Austin, in 2017. During Spring 2018, he was a visiting scholar with the Simons Institute at the University of California Berkeley. He is currently working towards the Ph.D. degree at UT Austin, under supervision of Prof. Evdokia Nikolova, where he is also affiliated with the Wireless Networking and Communications Group (WNCG). His research interest is in combinatorial optimization and approximation algorithms with focus on applications in signal processing, power systems and electricity market. He was a best paper award finalist at HICSS 2018, and received a best student paper award at ICASSP 2018.

Permanent address: ali.kh@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.